Hindawi
Scientific Programming
Volume 2022, Article ID 2809222, 12 pages
https://doi.org/10.1155/2022/2809222

Research Article

Discriminative Similarity-Balanced Online Hashing for Supervised Image Retrieval

Chen Chen,[1] **Xiaoqin Wang**,[1] **Xiao Chen**,[2] **Rushi Lan**,[1] **Zhenbing Liu**,[1] **and Xiaonan Luo**[1]

[1]*Guangxi Key Laboratory of Image and Graphic Intelligent Processing, Guilin University of Electronic Technology, Guilin 541004, China*
[2]*Ministry of Education Key Laboratory of Cognitive Radio and Information Processing, Guilin University of Electronic Technology, Guilin 541004, China*

Correspondence should be addressed to Rushi Lan; rslan2016@163.com

Received 26 October 2021; Revised 26 November 2021; Accepted 2 December 2021; Published 12 February 2022

Academic Editor: Le Sun

When virtualizing large-scale images of the real world, online hashing provides an efficient scheme for fast retrieval and compact storage. It converts high-dimensional streaming data into compact binary hash codes while saving the structural characteristics between samples into the Hamming space. Existing works usually update the hashing function based on the similarity between input data, or design a codebook to assign code words for each single input sample. However, assigning code words to multiple samples while retaining the balanced similarity of the image instances is still challenging. To address this issue, we propose a novel discriminative similarity-balanced online hashing (DSBOH) framework in this work. In particular, we first obtain the Hadamard codebook that guides the generation of discriminative binary codes according to label information. Then, we maintain the correlation between the new data and the previously arrived data by the balanced similarity matrix, which is also generated by semantic information. Finally, we joined the Hadamard codebook and the balanced similarity matrix into a unified hashing function to simultaneously maintain discrimination and balanced similarity. The proposed method is optimized by an alternating optimization technique. Extensive experiments on the CIFAR-10, MNIST, and Places205 datasets demonstrate that our proposed DSBOH performs better than several state-of-the-art online hashing methods in terms of effectiveness and efficiency.

1. Introduction

With the widespread use of digital monitoring facilities and the Internet, the generated streaming data have also increased correspondingly [1–4]. The processing of streaming data needs to be performed in approximately real time, which is very difficult for high-dimensional multimedia data such as images and videos [5, 6]. Online hashing can encode high-dimensional streaming data that arrive online into compact binary codes with low storage and efficient computation [7, 8]. In particular, it preserves the relationship among the samples into the Hamming space and updates the hashing function in the light of the newly arrived data to adapt to the new data instance [9, 10]. In view of the advantages of low storage and efficient computation, online hashing is widely applied in education, finance, military, among other industries [11–14].

Most existing online hashing methods have been devoted to the trade-off between accuracy and efficiency [15–17]. According to the learning strategy, people divide these techniques into unsupervised online hashing and supervised online hashing [18–20]. The well-known unsupervised methods mainly include online sketch hashing (SketchHash) [21], FasteR online sketch hash (FROSH) [22], and zero-mean sketch [23]. SketchHash designs the hashing function with the sketch scheme [21]. FROSH uses the independent subsampling random Hadamard transform on various small data blocks to get a compact and accurate sketch while speeding up the sketching procedure [22]. The zero-mean sketch method solves the uncertainty problem of

the offset value and improves the data processing efficiency by zero-mean sketch [23]. Supervised methods obtain better performance than unsupervised methods in most instances because of the utilization of label information. Some representative works include online hashing (OKH) [24, 25], adaptive hashing (AdaptHash) [26], online supervised hashing (OSH) [27, 28], online hashing with mutual information (MIHash) [29], balanced similarity for online discrete hashing (BSODH) [30], and Hadamard codebook-based online hashing (HCOH) [31]. These methods have achieved satisfactory performance.

However, some existing supervised online hashing methods still achieve unsatisfactory accuracy in real applications as they ignore any discriminative and balanced similarity. More specifically, HCOH generates discriminative binary codes with maximum information entropy by the Hadamard codebook, but ignores the local neighbor relationship among samples and only processes a single input. On the other hand, BSODH only considers balanced similarity based on the pairwise relationship and neglects the global data distribution, which results in a decrease in accuracy [32, 33]. Hence, both HCOH and BSODH have problems when applied to real applications.

In this work, we put forward a novel discriminative similarity-balanced online hashing (DSBOH) framework, which can simultaneously preserve the global distribution information of data and pairwise relationships between samples to generate discriminate hash codes with maximum information entropy. In particular, first, we maintain the maximum information entropy of hash codes via a Hadamard codebook. Then, the pairwise similarity matrix is adjusted to ensure that the updated scheme of balanced hash codes is used to preserve the correlation between the new and existing data. Finally, we combine the above attributes into a unified hashing function. An alternating iterative algorithm is used to solve the proposed DSBOH method. Compared with several state-of-the-art online hashing techniques, remarkable results have been achieved by our proposed DSBOH method.

In summary, the main contributions of this work include the following:

(i) The Hadamard matrix is used to ensure that the hash codes with maximum information entropy are separable and can deal with situations with unknown number of categories.

(ii) We preserve the balanced similarity between newly arrived data and previously arrived data into the generated Hamming space using the inner product to deal with uneven data distribution.

(iii) We combine the Hadamard codebook and the balanced similarity matrix into a unified hashing function to simultaneously maintain the discrimination and balanced similarity of the hashing modal.

(iv) The alternating iterative algorithm is used to optimize the proposed method, and experimental results verify that our method performs much better than several state-of-the-art online hashing techniques.

The remainder of this study is organized as follows. Section 2 gives a brief overview of the related works. In Section 3, we elaborate on the framework and optimization of the proposed method. Section 4 details the experimental results and analyses. Finally, we conclude the paper in Section 5.

2. Related Work

In this section, we present supervised methods, such as OKH [24, 25], OSH [27, 28], AdaptHash [26], MIHash [29], and BSODH [30].

Huang et al. first proposed a prototype based on online hashing termed OKH [24]. In each current iteration, a new pair of data samples is used, a pair of sample similarity loss functions is designed according to the Hamming distance, and the prediction loss referring to Ref. [34] is used. The function evaluates whether the operating hashing projection vector suits the new data and expects the model to save as much of the historical information of the previous round of projection vectors as possible during the update process. To make the original online hashing algorithm more perfect in loss function theory, an improved weakly supervised online hashing learning model [25], which does not require the label information of the data, is proposed for the loss threshold of the hashing modal. The new objective function is designed to calculate the disparity between the Hamming distances of pairwise data, and the upper limit loss of the online hash theory is rigorously analyzed. Second, because the hashing function learned in the algorithm update relies on new data, it easily falls into local deviations according to the characteristics of online hash algorithms to adapt to the new data; a multimodal strategy is produced to reduce such deviations.

Cakir et al. proposed the OSH method [27], which adapts to data changes, and the label types of datasets are unknown. A random method is used to generate the codebook, so that the code generated by the hashing function and the category matching error in the corresponding codebook are minimized [35]. To ensure the last round of information, the previous hashing functions are linearly combined and superimposed; however, the codebook structure directly determines the coding efficiency. Therefore, in the follow-up literature, an improved online supervised hashing [28] is proposed for this problem. The ECOC codebook is applied according to online supervised hashing, which improves the space efficiency and solves the original Hamming loss formula. Complexity proposes an efficient solution method based on the upper boundary, which improves the time efficiency of the algorithm.

AdaptHash [26] uses the relationship between data sample similarity and Hamming distance to solve the problem of how online models adapt to current data. First, the objective function is constructed using the similarity of current sample pairs and the Hamming distance relationship combined with the minimum loss variance function [36], and the gradient descent algorithm is used to solve the hash projection vector; the objective function is further

generalized to make similar or unsimilar sample data pairs. The Hamming distance is minimized (maximized) to reduce the update redundancy caused by the update mechanism; finally, the hinge loss function [37] is used to filter the hash map with the largest error, and the iterative calculation is reentered until the number of iterations reaches the set value.

MIHash [29] adopts the theory of quantitative information coding to obtain high-quality hash code that eliminates unnecessary hash table updates. The mutual information between the dataset samples is well correlated with standard evaluation indicators and is used to calculate the information entropy. When optimizing the mutual information target, differentiable histogram merging technology is used to derive stochastic gradient descent-based optimization rules, and finally, the differentiated rules are utilized to merge the derived histograms and apply them to the learning objective function. This work is dedicated to the synchronization of the hash code and the hashing function updates and effectively reduces the reconstruction of the hash table.

BSODH [30] studies the relationship between new data and previously arrived data. This work considers that the problem of online hashing is attributed to two issues: updating imbalance and optimization inefficiency. The above authors recommend asymmetric graph regularization techniques to keep the relevance of online streaming data and previously accumulated datasets. To deal with data imbalance in the learning stage of online hashing, BSODH designs a new balanced similarity matrix between new data and previously arrived data, which tackles the challenge of quantization error brought by relaxation learning in the discrete optimization method in online learning and reveals advanced results compared with the quantization-based schemes.

In addition, some existing offline deep hashing methods [38–42] use deep learning techniques to train the hashing function and map the image data into low-dimensional binary codes to complete the mission of image retrieval, but as the amount of data increases, the retraining model consumes more time whenever new data arrive. For example, deep transfer hashing (DTH) [42] trains a CNN model and inputs the online generated image pairs and their labels into the network. The loss function of the model makes the outputs of similar instances close, while the outputs of dissimilar instances are pushed farther, thus obtaining the binary codes representing the semantic structure of the original image pairs. However, this method requires a complex relaxation process and a relatively large number of bits to obtain satisfactory retrieval results.

3. The Proposed Framework

Figure 1 shows the overall framework of our proposed discriminative similarity-balanced online hashing (DSBOH), which contains two main modules, namely discriminative codebook and balanced similarity. The details of the proposed DSBOH are presented as follows.

3.1. Notations. Assume that N d-dimensional data denoted as $X^t = [x_1^t, x_2^t, \ldots, x_{n_t}^t] \in R^{d\times n_t}$ are fed into the system at the t stage, whose corresponding label L is expressed as $L^t = [l_1^t, l_2^t, \ldots, l_{n_t}^t] \in N^{n_t}$. Our goal is to generate k-dimensional binary codes $B^t = [b_1^t, b_2^t, \ldots, b_{n_t}^t] \in \{1, -1\}^{k\times n_t}$, $k \ll d$. The mapping matrix to be learned for reducing the d-dimensional real-valued data X^t to k-dimensional binary data B^t is represented as $W^t \in R^{d\times k}$. The expression of B^t is defined as follows:

$$B^t = F\left(X^t\right) = \operatorname{sgn}\left(W^{tT}X^t\right), \tag{1}$$

where $F(\cdot)$ represents the hashing function, W^{tT} represents the transposition of W^t, and $\operatorname{sgn}(\cdot)$ is the symbolic function defined as follows:

$$\operatorname{sgn}(x) = \begin{cases} 1, & 0 \le x, \\ -1, & 0 > x. \end{cases} \tag{2}$$

To retain the similarity or dissimilarity relationship between the newly arrived streaming data and the previously arrived data, we consider constructing the hashing function with a similarity matrix. At the t stage, the currently arriving data are defined as $X_c^t = [x_{c1}^t, x_{c2}^t, \ldots, x_{cn_t}^t]$ whose corresponding labels are represented as L_c^t, and the generated hash codes are represented as B_c^t. The data arriving before the t stage are $X_a^t = [X_c^1, X_c^2, \ldots, X_c^{t-1}]$ whose corresponding labels are represented as L_a^t, and the generated hash codes are represented as B_a^t. All symbol notations utilized in this study are presented in Table 1.

3.2. Hadamard Codebook. To maintain the maximum information entropy of hash codes, we construct a Hadamard codebook in three steps. First, we generate an orthogonal Hadamard matrix that is 2^q-dimensional (q is a positive integer) according to the definition $C_{ij} = (-1)^{(i-1)(j-1)}$, where C_{ij} is the jth element of the ith row in matrix C. The Hadamard matrix can generate independent hash codes that satisfy two principles of the error-correcting output code: the Hamming distance between columns is maximized to ensure a significant difference between classifiers, and the Hamming distance between rows is maximized to have a strong error correction ability. Attention should be paid to guarantee that the dimension of the Hadamard matrix is a bit larger than the number of labels. Second, we assign data from the same class to the same column vector of the Hadamard matrix C to be the target vector in the Hadamard codebook $\widetilde{C}$. In particular, when a batch of new data is received, we randomly and nonrepeatedly select certain columns in the Hadamard matrix to construct virtual multilabel vectors in the Hadamard codebook. When the label of the new data is the same as the data that arrived before, it is assigned to the same column vector. These vectors are aggregated to form a codebook $\widetilde{C}$. Finally, we use locality-sensitive hashing (LSH) [43] to align the code length of the Hadamard codebook with that of the hash codes.

To maintain the independence of the hash code and retain the global distribution information, we define the loss function L_1 based on the Hadamard codebook as follows:

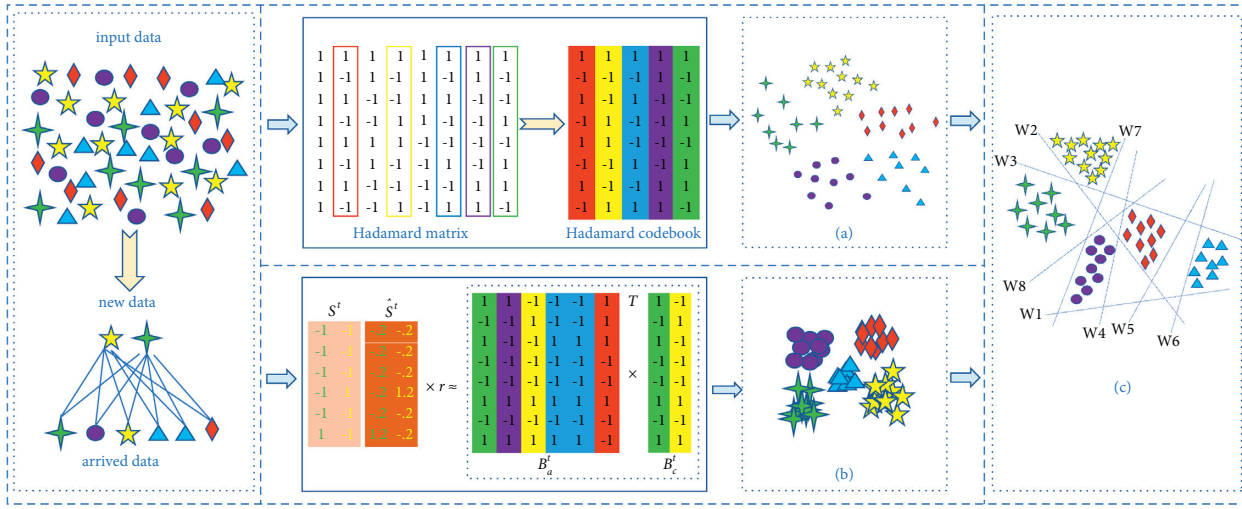

Figure 1: The overall framework of the proposed discriminative similarity-balanced online hashing (DSBOH). (a) The hash codes generated by the constructed Hadamard codebook for the input data are independent of each other, but the similar relationship between the data is ignored. (b) The similarity matrix constructed from the new data and the arrived data guides the generated hash codes to have a stronger classification ability but ignores the global data distribution. (c) The proposed algorithm can generate more discriminative hash codes for satisfactory retrieval results.

Table 1: Notations utilized in this study.

Symbol	Notations
X^t	Input data at t stage
L^t	Label of X^t
B^t	Binary codes generated for X^t
W^t	Hashing mapping matrix at t stage
X_c^t	Data arriving currently at t stage
L_c^t	Label of X_c^t
B_c^t	Binary codes generated for X_c^t
X_a^t	Data all arriving before t stage
L_a^t	Label of X_a^t
B_a^t	Binary codes generated for X_a^t
d	Dimension of input data
k	Dimension of binary code
N	Number of input data
n_t	Number of input data at t stage

$$L_1 = \min_{W^t} \left\| F\left(X^t\right) - \widetilde{C}_{J_{(X^t)}} \right\|_F^2, \tag{3}$$

where $\widetilde{C}_i$ represents the ith column of codebook $\widetilde{C}$, $J_{(x_i^t)}$ denotes the label category of x_i^t, and $\|\cdot\|_F$ is the Frobenius norm of a matrix.

3.3. Balanced Similarity. Suppose that there are two input data x_i and x_j, the corresponding labels are l_i and l_j and the hash codes are expressed as $B_i = [b_{i1}, b_{i2}, \ldots, b_{ik}]^T \in \{1, -1\}^{k\times 1}$ and $B_j = [b_{j1}, b_{j2}, \ldots, b_{jk}]^T \in \{1, -1\}^{k\times 1}$, respectively. S_{ij} represents the similarity matrix of x_i and x_j. If x_i and x_j belong to one category, that is, $l_i = l_j$, then $S_{ij} = 1$. We expect that the hash codes within the same category are the same; that is, $B_i = [b_{i1}, b_{i2}, \ldots, b_{ik}]^T = [b_{j1}, b_{j2}, \ldots, b_{jk}]^T = B_j$. Because the product of the same binary codes is 1, $B_j^T B_j = k = kS_{ij}$. Conversely, if x_i and x_j are within different categories, that is, $l_i \neq l_j$, then $S_{ij} = -1$. We also expect that the hash codes from different categories are different; that is, $B_i = [b_{i1}, b_{i2}, \ldots, b_{ik}]^T = [-b_{j1}, -b_{j2}, \ldots, -b_{jk}]^T = -B_j$. Because the product result of the different binary codes is -1, $B_i^T B_j = -k = kS_{ij}$. In sum, the product of B_i^T and B_j has a common value with kS_{ij}, which means that we can retain the similarity relationship of input data into the Hamming space through the above method as follows:

$$\min_{B_i^t, B_j^t} \left\| B_i^{tT} B_j^t - kS_{ij}^t \right\|_F^2. \tag{4}$$

To keep the similarity relationship constructed from the newly arrived data X_c^t at the t stage and the data X_a^t before the t stage in the Hamming space, the relationship between the inner product of the binary codes B_c^t and B_a^t and similarity S^t is used. In addition, with the increase in the new instances, the similarity matrix S^t becomes more and more sparse because most image pairs are dissimilar [30]. To prevent the model from overly relying on dissimilar information and ignoring the information of similar pairs, we adjust the similarity matrix according to the similarity and dissimilarity and convert the similarity matrix S^t to the balanced similarity matrix $\widetilde{S}^t$ by multiplying by different balance factors. The balanced similarity matrix $\widetilde{S}^t$ is defined as follows:

$$\widetilde{S}_{ij}^t = \begin{cases} \mu_s S_{ij}^t, & S_{ij}^t = 1, \\ \mu_d S_{ij}^t, & S_{ij}^t = -1, \end{cases} \tag{5}$$

where $\widetilde{S}_{ij}^t$ and S_{ij}^t represent the element in the ith row and jth column of matrices $\widetilde{S}^t$ and S^t, respectively. μ_s denotes the impact factor of the similarity pairs, while μ_d denotes the impact factor of the dissimilarity pairs. When μ_s is greater than μ_d, the Hamming distance between similar pairs will decrease, while that between dissimilar pairs will increase. By adjusting the two balance factors, the problem of data

imbalance can be solved. Thus, the loss function of balanced similarity can be defined as follows:

$$L_2 = \min_{B_a^t, B_c^t} \left\| B_c^{tT} B_a^t - k\widetilde{S}^t \right\|_F^2, \quad s.t.\ B_c^t \in \{1,-1\}^{k\times n_t}, \quad B_a^t \in \{1,-1\}^{k\times m_t}, \tag{6}$$

where $m_t = \sum_{i=1}^{t-1} n_i$ denotes the total number of instances that arrived before t stage.

3.4. Overall Formulation. Different from HCOH and BSODH, which find the global data distribution or balanced similarity via a local neighbor relationship, DSBOH aims to generate discriminative binary codes for single or multiple inputs by preserving global distribution information with the help of Hadamard codebook and local pairwise relationship between the newly arrived data and the previously arrived data in a seamless framework. When the data explode, the modal still has a strong generalization ability because we consider retaining the semantic relationship between the data at different stages. Furthermore, hash codes are independent and discriminative due to the use of codebook. Therefore, we combine loss function L_1 of the Hadamard codebook hashing function in equation (3) and loss function L_2 of balanced similarity preservation in equation (6) into the same objective function, which is expressed as follows:

$$\min_{B_a^t, B_c^t, W^t} \left\| B_c^{tT} B_a^t - k\widetilde{S}^t \right\|_F^2 + \lambda^t \left\| F\left(X^t\right) - \widetilde{C}_{J_{(X^t)}} \right\|_F^2, \quad \text{s.t. } B_c^t \in \{1,-1\}^{k\times n_t}, \quad B_a^t \in \{1,-1\}^{k\times m_t}, \tag{7}$$

where λ^t is the parameter to control the importance.

To minimize the quantization error between learned hashing function $F(X^t)$ and the target hash code B^t, the quantized loss function is defined as follows:

$$\min_{W^t} \left\| F\left(X^t\right) - B^t \right\|_F^2. \tag{8}$$

Finally, adding equation (8) into (7), and adding the Frobenius norm of W^t as a regular term, the overall formulation is expressed as follows:

$$L = \min_{B_a^t, B_c^t, W^t} \left\| B_c^{tT} B_a^t - k\widetilde{S}^t \right\|_F^2 + \lambda^t \left\| F\left(X^t\right) - \widetilde{C}_{J_{(X^t)}} \right\|_F^2 + \sigma^t \left\| F\left(X^t\right) - B^t \right\|_F^2 + \varepsilon^t \left\| W^t \right\|_F^2, \quad \text{s.t. } B_c^t \in \{1,-1\}^{k\times n_t}, \quad B_a^t \in \{1,-1\}^{k\times m_t}, \tag{9}$$

where σ^t and ϵ^t are parameters to control the importance of each module.

3.5. Alternating Optimization. Owing to the discrete restrictions of the binary codes, the optimization problem of the variables in equation (9) is nonconvex [44, 45]. In this regard, an alternating optimization technique is adopted to deal with our proposed loss function L. That is, when a variable is updated, others are fixed as constants. The specific details of the implementation are introduced as follows.

(1) Solving W^t: fix B_a^t and B_c^t, so that the first term in equation (9) can be eliminated. The objective function becomes:

$$\min_{W^t} \lambda^t \left\| F\left(X^t\right) - \widetilde{C}_{J_{(X^t)}} \right\|_F^2 + \sigma^t \left\| F\left(X^t\right) - B_c^t \right\|_F^2 + \varepsilon^t \left\| W^t \right\|_F^2. \tag{10}$$

Replacing the formula $F(X^t) = \operatorname{sgn}(W^{tT} X^t)$ in equation (1) with $F(X^t) = \tanh(W^{tT} X^t)$ for optimization convenience, we obtain the following:

$$\min_{W^t} \left\| \tan h\left(W^{tT} X_c^t\right) - \widetilde{C}_{J_{\left(X_c^t\right)}} \right\|_F^2 + \sigma^t \left\| \tanh\left(W^{tT} X_c^t\right) - B_c^t \right\|_F^2 + \varepsilon^t \left\| W^t \right\|_F^2. \tag{11}$$

Using the formula of matrix A:

$$\|A\|_F = \sqrt{tr\left(A^T A\right)} = \sqrt{tr\left(AA^T\right)}. \tag{12}$$

We convert equation (11) into the form of the trace of the matrix as follows:

$$\min_{W^t} tr\left(\left(W^{tT} X_c^t - \widetilde{C}_{J_{\left(X_c^t\right)}}\right)\left(X_c^{tT} W^t - \widetilde{C}_{J_{\left(X_c^t\right)}}^T\right)\right) + \sigma^t tr\left(\left(W^{tT} X_c^t - B_c^t\right)\left(X_c^{tT} W^t - B_c^{tT}\right)\right) + \varepsilon^t tr\left(W^t W^{tT}\right). \tag{13}$$

After simplification, we obtain the following:

$$\min_{W^t}\left[\left(1+\sigma^t\right)X_c^tX_c^{tT}+\int^t I\right]tr\left(W^tW^{tT}\right) - 2tr\left(W^{tT}X_c^t\left(\widetilde{C}_{J_{\left(X_c^t\right)}}+\sigma^tB_c^t\right)\right), \tag{14}$$

where I stands for the d-dimensional identity matrix. Equation (14) takes the partial derivative of W^t and makes the result zero. That is:

$$\left[\left(1+\sigma^t\right)X_c^tX_c^{tT}+\varepsilon^tI\right]W^t-X_c^t\left(\widetilde{C}_{J_{\left(X_c^t\right)}}+\sigma^tB_c^t\right)=0. \tag{15}$$

Therefore, we update W^t with the following equation:

$$W^t=\left[\left(1+\sigma^t\right)X_c^tX_c^{tT}+\varepsilon^tI\right]^{-1}X_c^t\left(\widetilde{C}^T_{J_{\left(X^t\right)}}+\sigma^tB_c^{tT}\right). \tag{16}$$

(2) Solving B_a^t: fix W^t and B_c^t; therefore, only the first term remains in equation (9). The objective function now becomes:

$$\min_{B_a^t,B_c^t,W^t}\left\|B_c^{tT}B_a^t-k\widetilde{S}^t\right\|_F^2. \tag{17}$$

According to Ref. [46], the F norm is changed to the $L1$ norm; the result is as follows:

$$B_a^t=\operatorname{sgn}\left(B_c^t\widetilde{S}^t\right). \tag{18}$$

(3) Solving B_c^t: fix W^t and B_a^t. Equation (9) becomes:

$$\min_{B_c^t}\left\|B_c^{tT}B_a^t-k\widetilde{S}^t\right\|_F^2+\sigma^t\left\|F\left(X^t\right)-B_s^t\right\|_F^2. \tag{19}$$

For further optimization, we remove irrelevant items and obtain the following:

$$\min_{B_c^t}\left\|B_c^{tT}B_a^t\right\|_F^2-2tr\left(P^TB_s^T\right), \tag{20}$$

where $P=k\lambda^tB_a^t\widetilde{S}^{tT}+\sigma^tW^{tT}X_c^t$. According to supervised discrete hashing (SDH) [6] and BSODH [30], the optimization in equation (20) is NP hard, so we turn the matrix into a combination of row vectors, transferring the problem into row by row updating. That is to say, equation (20) becomes:

$$\min_{\widetilde{b}_{cr}^t}\left\|\widetilde{b}_{ar}^{tT}\widetilde{b}_{cr}^t+\widetilde{B}_a^{t\,T}\widetilde{B}_c^t\right\|_F^2-2tr\left(\widetilde{p_{ar}^t}^T\widetilde{b}_{cr}^t+\widetilde{P}^{t\,T}\widetilde{B}_c^t\right), \tag{21}$$

where $\widetilde{b}_{cr}^t$, $\widetilde{b}_{ar}^t$, and $\widetilde{p}_r$ are the rth row of B_c^t, B_a^t, and P; $\widetilde{B}_c^t$, $\widetilde{B}_a^t$, and $\widetilde{P}^t$ are the remaining parts of B_c^t, B_a^t, and P except for the rth row, respectively. The above formula is expanded to obtain the following:

$$\min_{\widetilde{b}_{cr}^t}\left\|\widetilde{b_{ar}^t}^T\widetilde{b}_{cr}^t\right\|_F^2+\left\|\widetilde{B}_a^{t\,T}\widetilde{B}_c^t\right\|_F^2+2tr\left(\widetilde{B}_c^{t\,T}\widetilde{B}_a^t\widetilde{b_{ar}^t}^T\right) - 2tr\left(\widetilde{p_{ar}^t}^T\widetilde{b}_{cr}^t\right)-2tr\left(\widetilde{P}^{t\,T}\widetilde{B}_c^t\right), \tag{22}$$

The equation (22) is simplified to obtain the following:

$$\min_{\widetilde{b}_{cr}^t}tr\left(\left(\widetilde{B}_c^{t\,T}\widetilde{B}_a^t\widetilde{b_{ar}^t}^T-\widetilde{p_r^t}^T\right)\widetilde{b}_{cr}^t\right). \tag{23}$$

Therefore, we update row by row according to the following rules:

$$\widetilde{b}_{cr}^t=\operatorname{sgn}\left(\widetilde{p}_r-\widetilde{b}_{ar}^t\widetilde{B}_a^{t\,T}\widetilde{B}_c^t\right). \tag{24}$$

The proposed DSBOH is summarized in Algorithm 1.

Input: training instances X; labels L; the number of data batches T; code length k; parameters $\lambda^t, \sigma^t, \epsilon^t$.

Output: binary codes B and hash map matrix W.

Initialize W and W_{LSH} with the normal Gaussian distribution

Generate Hadamard matrix of r-dimension

if $r \neq k$ **then**

Adopt LSH for Hadamard to get codebook C

else

Make Hadamard as codebook C

end if

while $T \leftarrow 1$ **do**

Denote the data coming currently as X_c^t

Set $X_a^t = [X_a^t; X_c^t]$, $B_a^t = [B_a^t; B_c^t]$

Compute S according to labels

Update W^t via equation (16) and B_a^t via equation (18)

while r becomes $k \leftarrow 1$ **do**

Update b_{cr}^t via equation (24)

end while

end while

Set $W = W^t$ and calculate $B^t = \operatorname{sgn}(W^{tT}X^t)$

Return W, B

4. Experiments

To prove the effectiveness of DSBOH, extensive experiments on three widely used image datasets are conducted in this section and compared them with several advanced online hashing techniques.

4.1. Datasets. CIFAR-10 [47] is an inclusively applied dataset for image retrieval and classification. It is composed of 60,000 samples selected from ten classes, and each sample

is represented by 4096-dimensional CNN features. The ten classes are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each category includes 6,000 samples. We randomly select 5,900 samples from each category as the training set; the remaining images are set as the testing set. From the training set, 20,000 instances are utilized for learning hashing functions [31]. Twenty example images from each category of CIFAR-10 are shown in Figure 2.

MNIST consists of 70,000 hand-written digital images with 10 categories, which include numbers 0 to 9; each image is represented by a 784-dimensional vector. We randomly sample 100 instances from each class to construct the testing set and make use of the remaining part to compose the training set. 20,000 images randomly selected from the training set are used to learn the hash model [18]. We randomly select 27 example instances from each class to show in Figure 3.

Places205 [48] is a large-scale scene-centric dataset that contains 205 common scene categories and 2.5 million images with labels. First, the fc-7 layer of AlexNet [49] calculates the features of each image, and then, PCA is exploited to simplify these features into 128-dimensional vectors. We stochastically choose 20 images from each category to form the test set, and the others automatically consist of the training set. 100,000 images in the training set are randomly selected to learn the hashing functions. Two hundred randomly picked images of Places205 are shown in Figure 4.

Figure 2: Example images of CIFAR-10 dataset.

Figure 3: Example images of MNIST dataset.

Figure 4: Example images of Places205 dataset.

4.2. Experimental Settings

4.2.1. Parameter Settings. According to experience, the ranges of λ^t, σ^t, and ϵ^t for the proposed DSBOH are set in {0: 0.05: 5}. For the CIFAR-10 dataset, the best combination for $(\lambda^t, \sigma^t, \epsilon^t)$ is empirically adopted to $(0.7, 0.3, 0.8)$. For the MNIST dataset, we set $(0.1, 0.3, 1.2)$ as the configuration of $(\lambda^t, \sigma^t, \epsilon^t)$. For the Places205 dataset, $(0.1, 0.8, 0.2)$ corresponds to $(\lambda^t, \sigma^t, \epsilon^t)$. Table 2 shows the detailed parameters of DSBOH on the CIFAR-10, MNIST, and Places205 datasets. In addition, we conducted experiments with hash codes of different lengths from the set [8, 16, 32, 48, 64, 128]. It is worth mentioning that SketchHash requires the size of a batch greater than that of hash codes [21]. Thence, we only show the results of SketchHash under 64 bits.

4.2.2. Evaluation Protocols. To evaluate the proposed method, we apply a set of widely adopted protocols, which includes the mean average precision (mAP), the average accuracy of the first 1000 retrieved samples (mAP@1000), which is used for the large dataset Places205 to reduce the calculation time, precision within a Hamming sphere with a radius of 2 centered on every query point (Precision@H2), and the average precision of top-R retrieving neighbors (Precision@R). We also compare the running time on CIFAR-10 and MNIST with other methods. Additionally, the precision-recall curves on CIFAR-10 and MNIST are adopted to evaluate our proposed method.

Table 2: Parameter settings to CIFAR-10, MNIST, and Places205 on DSBOH.

Parameter	CIFAR-10	MNIST	Places205
λ^t	0.7	0.1	0.1
σ^t	0.3	0.3	0.8
ϵ^t	0.8	1.2	1.2
μ_s	1.5	1.5	1.5
μ_d	0.5	0.5	0.5
n_t	2000	20 000	20 000

4.2.3. Compared Methods. We contrast the proposed DSBOH with several advanced online hashing methods, including OKH [25], OSH [28], AdaptHash [26], SketchHash [21], and BSODH [30]. All the results of the above methods are implemented via the publicly available source codes. We implement all the methods using MATLAB on a single computer equipped with a 3.0 GHz Intel Core i5-8500 CPU and 16 GB RAM; all results shown in this work are the average of the three runs.

Table 3: mAP and Precision@H2 results on CIFAR-10 for 8, 16, 32, 48, 64, and 128 bits.

Methods	mAP						PrecisionH2					
	8 bits	16 bits	32 bits	48 bits	64 bits	128 bits	8 bits	16 bits	32 bits	48 bits	64 bits	128 bits
OKH [25]	0.100	0.134	0.223	0.252	0.268	0.350	0.100	0.175	0.100	0.452	0.175	0.372
OSH [28]	0.123	0.126	0.129	0.131	0.127	0.125	0.120	0.123	0.137	0.117	0.083	0.038
AdaptHash [26]	0.116	0.138	0.216	0.297	0.305	0.293	0.114	0.254	0.185	0.093	0.166	0.164
SketchHash [21]	0.248	0.301	0.302	0.327	——	——	0.256	0.431	0.385	0.059	——	——
BSODH [30]	**0.564**	0.604	0.689	0.656	0.709	0.711	0.305	0.582	0.691	**0.697**	**0.690**	**0.602**
DSBOH	0.556	**0.669**	**0.703**	**0.696**	**0.720**	**0.727**	**0.411**	**0.730**	**0.737**	0.655	0.552	0.371

The first-ranked results are given in bold.

Table 4: mAP and Precision@H2 results on MNIST for 8, 16, 32, 48, 64, and 128 bits.

Methods	mAP						Precision@H2					
	8 bits	16 bits	32 bits	48 bits	64 bits	128 bits	8 bits	16 bits	32 bits	48 bits	64 bits	128 bits
OKH [25]	0.100	0.155	0.224	0.273	0.301	0.404	0.100	0.220	0.457	0.724	0.522	0.124
OSH [28]	0.130	0.144	0.130	0.148	0.146	0.143	0.131	0.146	0.192	0.134	0.109	0.019
AdaptHash [26]	0.138	0.207	0.319	0.318	0.292	0.208	0.153	0.442	0.535	0.335	0.163	0.168
SketchHash [21]	0.257	0.312	0.348	0.369	——	——	0.261	0.596	0.691	0.251	——	——
BSODH [30]	0.593	0.700	0.747	0.743	0.766	0.760	0.308	0.709	0.826	**0.804**	**0.814**	**0.643**
DSBOH	**0.596**	**0.721**	**0.759**	**0.751**	**0.781**	**0.781**	**0.403**	**0.803**	**0.849**	0.788	0.651	0.415

The first-ranked results are given in bold.

Table 5: mAP@1000 and Precision@H2 results on Places205 for 8, 16, 32, 48, 64, and 128 bits.

Methods	mAP@1000						Precision@H2					
	8 bits	16 bits	32 bits	48 bits	64 bits	128 bits	8 bits	16 bits	32 bits	48 bits	64 bits	128 bits
OKH [25]	0.018	0.033	0.122	0.048	0.114	0.258	0.007	0.010	0.026	0.017	**0.217**	0.075
OSH [28]	0.018	0.021	0.022	0.032	0.043	0.164	0.007	0.009	0.012	0.023	0.030	0.059
AdaptHash [26]	0.028	0.097	0.195	0.223	0.222	0.229	0.009	0.051	0.012	0.185	0.021	0.022
SketchHash [21]	**0.052**	0.120	0.202	0.242	——	——	**0.017**	0.066	0.220	0.176	——	——
BSODH [30]	0.035	**0.174**	**0.250**	0.273	0.308	0.337	0.009	**0.101**	0.241	**0.246**	0.212	**0.101**
DSBOH	0.046	0.154	0.249	**0.286**	**0.313**	**0.347**	0.011	0.089	**0.264**	0.175	0.119	0.037

The first-ranked results are given in bold.

4.3. Results and Discussion. First, we can observe the experimental results of mAP and Precision@H2 on CIFAR-10 in Table 3. From this table, we can find that (1) mAP: in the case of 16 bits, 32 bits, 48 bits, 64 bits, and 128 bits hash codes, our proposed method has improved the second-best BSODH method by 6.5%, 1.4%, 4.0%, 1.1%, and 1.6%, respectively, and the mAP of DSBOH is slightly lower than that of BSODH. (2) Precision@H2: in the case of 8 bits, 16 bits, and 32 bits, our proposed method is 10.6%, 14.8%, and 4.6% better than the second-best BSODH, respectively. Although the mAP at 48 bits, 64 bits, and 128 bits of our proposed DSBOH slightly decreases compared with BSODH, our DSBOH performs better than other online hashing methods.

Table 4 shows the mAP and Precision@H2 results of our raised DSBOH and compared techniques on the MNIST dataset. The consequences indicate that (1) mAP: the proposed DSBOH accomplishes an increase of 0.3%, 2.1%, 1.2%, 0.8%, 1.5%, and 2.1% for mAP compared with the second-best BSODH in 8 bits, 16 bits, 32 bits, 48 bits, 64 bits, and 128 bits. Hence, the superiorities of DSBOH are demonstrated. (2) Precision@H2: the Precision@H2 of our DSBOH is much better than BSODH by 9.5%, 9.4%, and 2.3% for the 8 bits, 16 bits, and 32 bits, respectively. The performance of our DSBOH is slightly lower than that of BSODH in terms of 48 bits, 64 bits, and 128 bits.

The experimental consequences of mAP@1000 and Precision@H2 on the Places205 database are expressed in Table 5. From this table, we can learn that (1) mAP@1000: our proposed DSBOH is 1.3%, 0.5%, and 1.0% better than the second-best BSODH in terms of 48 bits, 64 bits, and 128 bits, respectively, and ranks second in terms of 8 bits, 16 bits, and 32 bits. (2) Precision@H2: the outcome of Precision@H2 for our DSBOH is the highest at 32 bits and 2.3% higher than the second-best method. For other hash bit lengths, DSBOH slightly decreases compared with the best.

For further verification of the performance of our DSBOH, we execute comparative experiments on Precision@R under 16 bits, 32 bits, and 64 bits hash codes on the CIFAR-10 and MNIST datasets. As shown in Figure 5, the proposed approach continuously reveals the best Precision@R, which demonstrates the superiority of DSBOH. In addition, the precision-recall curves on the CIFAR-10 and MNIST datasets are shown in Figures 6(a) and 6(b), respectively. Both curves wrap more curves, which proves the effectiveness of our algorithm. To clearly show the performance, we calculate the blue area under the curve (AUC) of the PR curves on CIFAR-10 and MNIST and obtain 95.70%

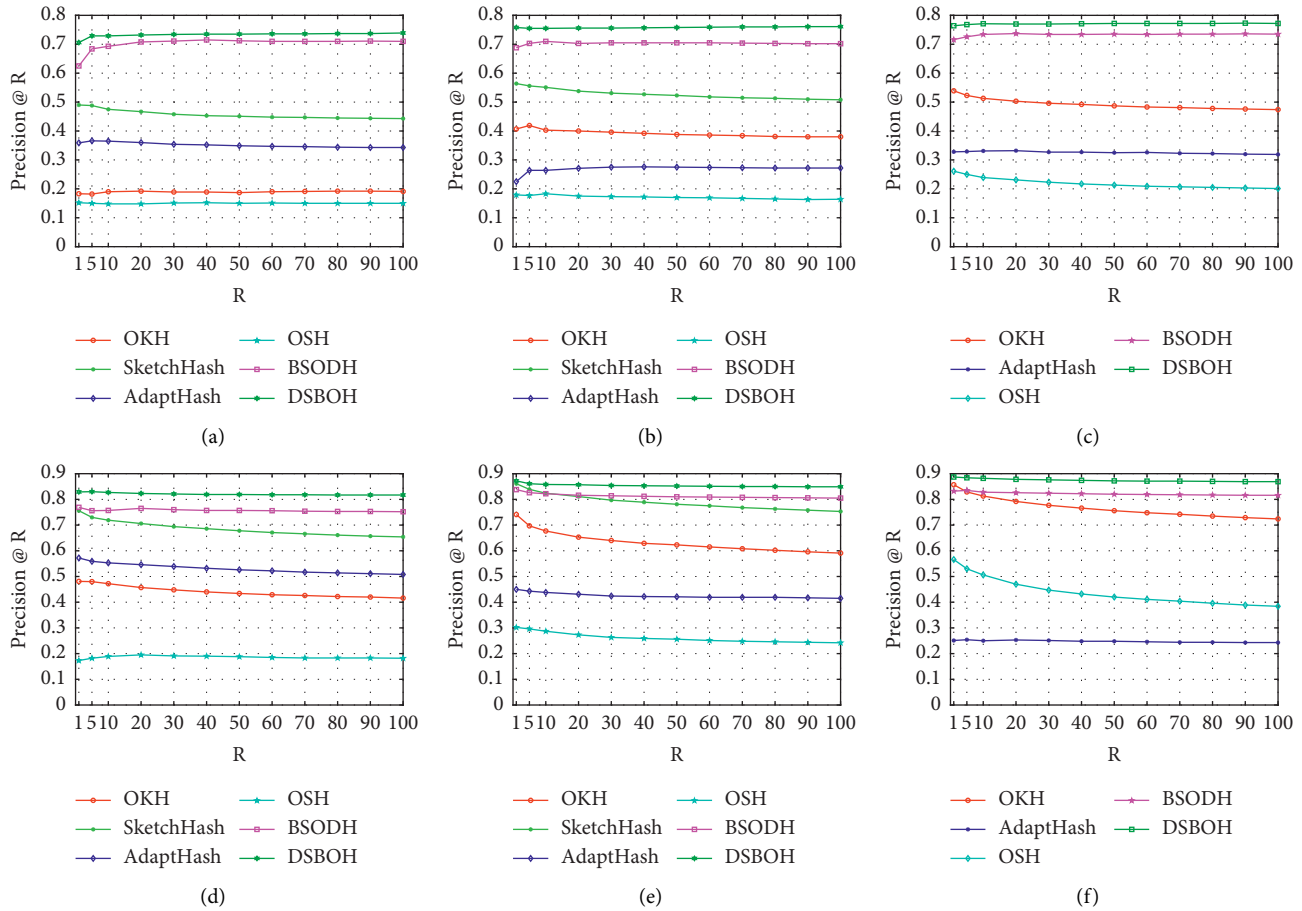

Figure 5: Precision@R curves of compared algorithms on CIFAR-10 and MNIST. (a) 16 hash bits on CIFAR-10, (b) 32 hash bits on CIFAR-10, (c) 64 hash bits on CIFAR-10, (d) 16 hash bits on MNIST, (e) 32 hash bits on MNIST, and (f) 64 hash bits on MNIST.

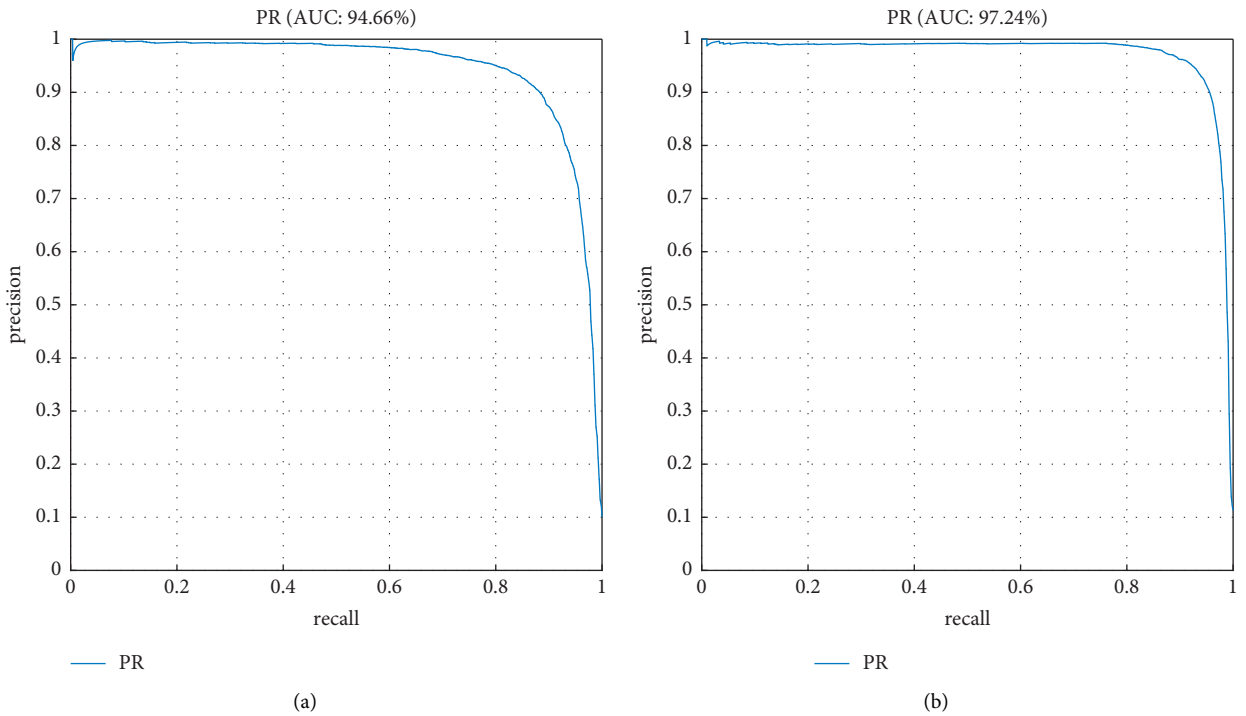

Figure 6: Precision-recall curve on CIFAR-10 and MNIST under 32 bit hashing codes. (a) PR curve on CIFAR-10 and (b) PR curve on MNIST.

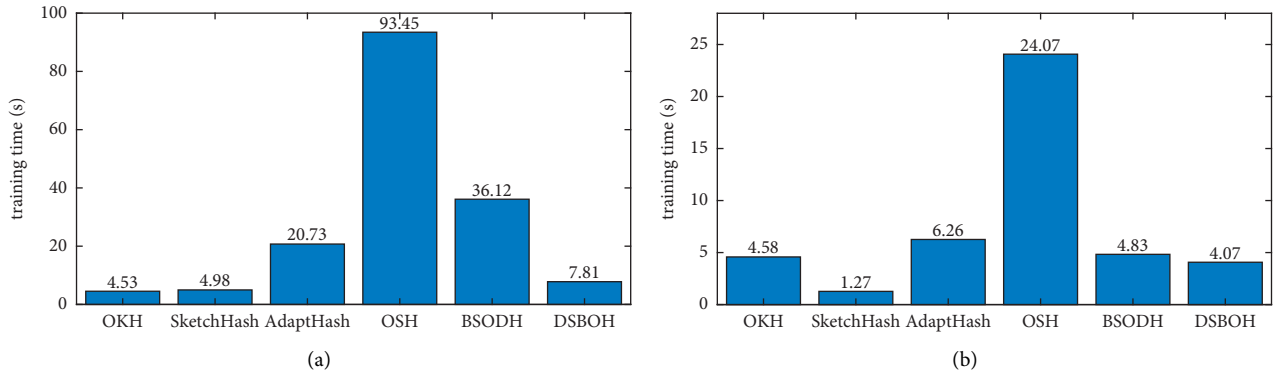

Figure 7: Training time of compared methods on CIFAR-10 and MNIST under 32 bit hashing codes. (a) Training time of compared methods on CIFAR-10 and (b) training time of compared methods on MNIST.

and 97.28% AUCs, respectively, which verifies that our learning model has a double high ratio of precision and recall.

4.4. Training Efficiency. Figure 7 presents the training time of our proposed method and compared approaches in terms of 32 bits on the CIFAR-10 dataset and MNIST dataset. As for Figure 7(a), we notice that our proposed DSBOH runs faster than AdaptHash, OSH, and BSODH but is very similar to OKH and SketchHash. We find that in CIFAR-10, although OKH and SketchHash have the shortest training time, their model accuracy is very poor. The training time spent by DSBOH is the shortest among the remaining algorithms, and the training efficiency is the highest. According to Figure 7(b), the training time of every method for comparison exceeds our proposed DSBOH except for SketchHash. Therefore, our algorithm is efficient for online image retrieval.

5. Conclusion

In this study, we bring forward DSBOH as a novel scheme that combines global distribution and balanced similarity to generate discriminative hash codes for image retrieval. To this end, we utilize the Hadamard codebook to assist the construction of the hashing function and keep the similarity between the newly arrived samples and the previously arrived samples from the original real value space into the Hamming space. Vast experiments on three benchmark datasets demonstrated that DSBOH shows significant advantages in effectiveness and efficiency compared with several innovatory online hashing methods. Since we use the codebook to assign code words to single-label images, the problem of code word assignment applied to multilabel image retrieval is worthy of further study. It is also possible to study a codebook that can better store the structure information of the image data in the future.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the National Natural Science Foundation of China (nos. 62 172 120, 61762028, and 61 962 014), Guangxi Science and Technology Project (nos. AD18281079, 2019GXNSFFA245014, 2019AC20014, and AB19110038), Ministry of Education Key Laboratory of Cognitive Radio and Information Processing (no. CRKL190109), and Guangxi Key Laboratory of Intelligent Processing Computer Image and Graphics (no. GIIP2001).

References

[1] H. Lu, R. Yang, Z. Deng, Y. Zhang, G. Gao, and R. Lan, "Chinese image captioning via fuzzy attention-based densenet-bilstm," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 17, no. 1, 2021.

[2] H. Liu, R. Ji, J. Wang, and C. Shen, "Ordinal constraint binary coding for approximate nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 4, pp. 941–955, 2019.

[3] H. Wang, X. Liu, and X. Nie, "Supervised discrete hashing through similarity learning," *Multimedia Tools and Applications*, vol. 80, no. 11, Article ID 16215, 2021.

[4] S. Ge, C. Li, S. Zhao, and D. Zeng, "Occluded face recognition in the wild by identity-diversity inpainting," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 30, no. 10, pp. 3387–3397, 2020.

[5] X. Wang, R. Lan, H. Wang, Z. Liu, and X. Luo, "Fine-grained correlation analysis for medical image retrieval," *Computers & Electrical Engineering*, vol. 90, Article ID 106992, 2021.

[6] 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)*, pp. 37–45, Boston, MA, USA, June 2015.

[7] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing: binary code embedding with hyperspheres," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2304–2316, 2015.

[8] M. Hashemi and M. Hall, "Detecting and classifying online dark visual propaganda," *Image and Vision Computing*, vol. 89, pp. 95–105, 2019.

[9] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 99, p. 1, 2016.

[10] Z. Chen, H. Lu, S. Tian et al., "Construction of a hierarchical feature enhancement network and its application in fault recognition," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 7, pp. 4827–4836, 2021.

[11] P. Wang, D. Wang, X. Zhang et al., "Numerical and experimental study on the maneuverability of an active propeller control based wave glider," *Applied Ocean Research*, vol. 104, 2020.

[12] X. Zhang, X. Li, Y. Liu, and F. Feng, "A survey on freehand sketch recognition and retrieval," *Image and Vision Computing*, vol. 89, pp. 67–87, 2019.

[13] R. Lan, Y. Zhou, Z. Liu, and X. Luo, "Prior knowledge-based probabilistic collaborative representation for visual recognition," *IEEE Transactions on Cybernetics*, vol. 50, no. 4, pp. 1498–1508, 2020.

[14] S. Ge, Z. Luo, C. Zhang, Y. Hua, and D. Tao, "Distilling channels for efficient deep tracking," *IEEE Transactions on Image Processing*, vol. 29, pp. 2610–2621, 2020.

[15] Z. Liu, F. Chen, and S. Duan, "Distributed fast supervised discrete hashing," *IEEE Access*, vol. 7, 2019.

[16] P. K. Agarwal and R. Sharathkumar, "Streaming algorithms for extent problems in high dimensions," *Algorithmica*, vol. 72, no. 1, pp. 83–98, 2015.

[17] S. Hong, H. Park, J.-S. No, T. Helleseth, and Y.-S. Kim, "Near-optimal partial Hadamard codebook construction using binary sequences obtained from quadratic residue mapping," *IEEE Transactions on Information Theory*, vol. 60, no. 6, pp. 3698–3705, 2014.

[18] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. 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.

[19] U. Chaudhuri, B. Banerjee, A. Bhattacharya, and M. Datcu, "Crossatnet–a novel cross-attention based framework for sketch-based image retrieval," *Image and Vision Computing*, vol. 104, 2020.

[20] J. Tan, T. Zhang, L. Zhao, X. Luo, and Y. Y. Tang, "A robust image representation method against illumination and occlusion variations," *Image and Vision Computing*, vol. 112, 2021.

[21] C. Leng, J. Wu, J. Cheng, X. Bai, and H. Lu, "Online sketching hashing," in *Proceedings of the IEEE Conference on Computer Vision Pattern Recognition*, pp. 2503–2511, Boston, MA, USA, June 2015.

[22] X. Chen, I. King, and M. R. Lyu, "Frosh: faster online sketching hashing," in *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017*, E. Gal, K. Kersting, and A. T. Ihler, Eds., pp. 1790–2022pp. 1790–, Sydney, Australia, August 2017.

[23] Y. Long, L. Liu, S. Fumin, L. Shao, and L. Xuelong, "Zero-shot learning using synthesised unseen visual data with diffusion regularisation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, p. 99, 2017.

[24] L.-K. Huang, Q. Yang, and W.-S. Zheng, "Online hashing," in *Proceedings of the IJCAI International Joint Conference on Artificial Intelligence*, pp. 1422–1428, Beijing, China, August 2013.

[25] L.-K. Huang, Q. Yang, and W.-S. Zheng, "Online hashing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2309–2322, 2018.

[26] F. Cakir and S. Sclaroff, "Adaptive hashing for fast similarity search," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1044–1052, Santiago, Chile, December 2015.

[27] F. Cakir and S. Sclaroff, "Online supervised hashing," in *Proceedings of the International Conference on Image Processing, ICIP*, pp. 2606–2610, Quebec City, Canada, September 2015.

[28] F. Cakir, S. A. Bargal, and S. Sclaroff, "Online supervised hashing," *Computer Vision and Image Understanding*, vol. 156, pp. 162–173, 2017.

[29] F. Cakir, K. He, S. A. Bargal, and S. S. Mihash, "Online hashing with mutual information," in *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 437–445, Venice, Italy, March 2017.

[30] M. Lin, R. Ji, H. Liu, X. Sun, Y. Wu, and Y. Wu, "Towards optimal discrete online hashing with balanced similarity," in *Proceedings of the 33rd AAAI Conference on Artificial Intelligence, AAAI*, pp. 8722–8729, Honolulu, Hawai, February 2019.

[31] M. Lin, R. Ji, H. Liu, and Y. Wu, "Supervised online hashing via Hadamard codebook learning," in *Proceedings of the 2018 ACM Multimedia Conference*, pp. 1635–1643, Seoul, Korea, October 2018.

[32] D. Wang, Q. Wang, Y. An, X. Gao, and Y. Tian, "Online collective matrix factorization hashing for large-scale cross-media retrieval," in *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1409–1418, Virtual Event, China, July 2020.

[33] Z. Qian, J. Xu, K. Zheng, P. Zhao, and X. Zhou, "Semantic-aware top-k spatial keyword queries," *World Wide Web*, vol. 21, no. 3, pp. 573–594, 2018.

[34] D. Xu, I. W. Tsang, and Y. Zhang, "Online product quantization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 11, pp. 2185–2198, 2018.

[35] M. Liu, D. Zhang, S. Chen, and H. Xue, "Joint binary classifier learning for ecoc-based multi-class classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 11, pp. 2335–2341, 2016.

[36] M. Long, C. Yue, Z. Cao, J. Wang, and M. I. Jordan, "Transferable representation learning with deep adaptation networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 12, pp. 3071–3085, 2018.

[37] L.-W. Huang, Y.-H. Shao, J. Zhang, Y.-T. Zhao, and J.-Y. Teng, "Robust rescaled hinge loss twin support vector machine for imbalanced noisy classification," *IEEE Access*, vol. 7, Article ID 65390, 2019.

[38] L. Zhu, H. Cui, Z. Cheng, J. Li, and Z. Zhang, "Dual-level semantic transfer deep hashing for efficient social image retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 4, pp. 1478–1489, 2021.

[39] L. Liao and Z. Li, "Deep hashing using n-pair loss for image retrieval," in *Proceedings of the 16th International Conference on Computational Intelligence and Security (CIS)*, pp. 20–24, Minsk, Belarus, November 2020.

[40] J. Zhang and Y. Peng, "Query-adaptive image retrieval by deep-weighted hashing," *IEEE Transactions on Multimedia*, vol. 20, no. 9, pp. 2400–2414, 2018.

[41] A. K. Bhunia, P. S. Raj Kishore, P. Mukherjee, A. Das, and P. P. Roy, "Texture synthesis guided deep hashing for texture

image retrieval," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, pp. 609–618, Waikoloa Village, Hawaii, January 2019.

[42] H. Zhai, S. Lai, H. Jin, X. Qian, and T. Mei, "Deep transfer hashing for image retrieval," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31, no. 2, pp. 742–753, 2021.

[43] M. Datar, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the 20th ACM Symposium on Computational Geometry*, pp. 253–262, NewYork, NY, USA, June 2004.

[44] S. Shalev-Shwartz, "Online learning and online convex optimization," *Foundations and Trends in Machine Learning*, vol. 4, no. 2, 2011.

[45] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, 1998.

[46] W.-C. Kang, W.-J. Li, and Z.-H. Zhou, "Column sampling based discrete supervised hashing," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pp. 1230–1236, AAAI Press, Phoenix, AZ, USA, February 2016.

[47] A. Torralba, R. Fergus, and T. William, "Freeman. 80 million tiny images: a large data set for nonparametric object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.

[48] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba, "Places: a 10 million image database for scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, 2017.

[49] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.