SSRG International Journal of Computer Science and Engineering
ISSN: 2348 – 8387 / https://doi.org/10.14445/23488387/IJCSE-V10I1P101

Volume 10 Issue 1, 1-16, January 2023

Original Article

Label Projection based on Hadamard Codes for Online Hashing

Nannan Wu[1], Zhen Wang[1, 2, *], Xiaohan Yang[1], Wenhao Liu[1], Xinyi Chang[1], Dongrui Fan[1]

[1]*School of Computer Science and Technology, Shandong University of Technology, Zibo, Shandong, China.*
[2]*Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun, Jilin, China.*

Received: 20 November 2022 Revised: 23 December 2022 Accepted: 05 January 2023 Published: 17 January 2023

Abstract *- When new data streaming arrives, traditional hashing methods should retrain the hashing functions based on all samples. That leads to high training time complexity. In contrast, the online hashing algorithm re-computes the hashing functions just based on the new arrival streaming data and has been widely applied in large-scale image retrieval tasks. However, differences exist in numbers and labels between new arrival and old datasets, which causes the data imbalance problem while establishing their similarity matrix. This paper proposes a novel supervised online hashing method, Label Projection, based on Hadamard Codes for Online Hashing (LHOH), which jointly employs the label projection and similarity preservation mechanism to solve the data imbalance problem. In addition, LHOH considers the Hadamard codes as the label projection target domains to avoid the problem of difficult discrete optimization of the objective function. Then, LHOH employs the label projection matrix as label weight values, which can solve the data imbalance problem while computing the similarity matrix between new arrival and old datasets and preserve the consistency of Hamming and semantic space similarity. To increase the distinguishability among the hash codes, LHOH designs triple supervision learning mechanisms, including assigning Hadamard codes, projecting labels, and embedding labels. To validate the performance of the proposed LHOH method, this paper sets up the approximate nearest neighbor (ANN) search comparative experiments on two widely used datasets. The final results show that LHOH outperforms six current state-of-the-art online methods.*

Keywords *- Online hashing, Hadamard code, Label projection, Triple supervision, Image retrieval.*

1. Introduction

With the development of Internet technology, massive multimedia data streaming such as images, videos, and audio are flooding into the network. And they are characterized by diversity, rapid growth, and large quantity. For large-scale multimedia data, it has become a prevalent issue for fast response the multimedia retrieval request in the computer vision field.

Traditional hashing methods[1-5] map high-dimensional floating-point data into compact binary codes and preserve their original similarity relationship in the Hamming space.[6-9] Hash-based approximate nearest neighbor (ANN) search methods have been widely used to respond to large-scale multimedia ANN search tasks[10-12] due to their less storage space and fast computation. However, traditional hashing methods learn the hash function offline and should re-compute the hashing functions based on all data when new multimedia data arrives. Unfortunately, the internet dataset rapidly increases, which leads to the high training time complexity of offline learning procedures. Online hashing algorithms learn hashing functions based only on new data streaming to solve the above problem. However, data streaming has the characteristics of uncertainty, real-time, and novelty, which makes it a challenge to update hash functions online with high quality.

Recently, researchers proposed online hashing such as Online Kernel Hashing(OKH),[13] Adaptive Hashing (AdaptHash),[14] and Online Supervised Hashing (OSH)[15] to respond fast the real-time multimedia ANN search task. Online hashing often uses a sign function to map floating point data to binary codes. However, the sign function usually causes NP-hard problems. To avoid this problem, AdaptHash and Hadamard Codebook based on Online Hashing (HCOH)[16] transform the objective function solution space into an approximately continuous, which usually results in inefficient optimization.[17] With the development of discrete optimization in traditional hashing methods,[18] Some online hashing algorithms [17, 19, 20] propose to update the objective function using a discrete iterative scheme, which builds the foundation for the discrete optimization of online hashing.

In practical applications, the new and old datasets are significantly different in the quantity and content of the features. As a result, this causes a data imbalance problem and leads to asymmetric sparse semantic similarity matrices.[17] To solve the above problem, BSODH separately assigns different weight values to the similar and dissimilar sample pairs by utilizing two balancing factors. However, BSODH artificially sets the balancing factors, which may change the original distance between the new

and old datasets. LPOH maps labels to hash codes and considers the label projection matrix as weight values. Unfortunately, the target matrix is unknown, and the binary codes are discrete in LPOH. Its objective functions are hard to optimize.

This paper proposes Label projection based on Hadamard codes for Online Hashing(LHOH), which jointly exploits label projection and similarity preservation mechanism to solve the data imbalance problem. LHOH offline generates Hadamard codes based on the data label and utilizes them as the target domain for the label projection. Then, we can directly obtain the closed solution of the label projection matrix using a least square regression method. Afterwards, the label projection matrix is utilized as the label weight value. We compute the semantic similarity matrix between the new and old datasets based on the weight labels. As a result, we resolve the data imbalance problem of the semantic similarity matrix. In addition, LHOH establishes a triple supervision mechanism in the three processes: Hadamard encoding, label projection, and similarity matrix learning. This process utilizes labels as supervision information that enhances the performance of semantic similarity preservation in Hamming space.

2. Related Work

Online hashing updates the hash functions according to new data streaming, which can obtain superior ANN search performance on old and new datasets. With the advantages of low training complexity, insufficient storage space occupation, and flexible application scenarios, more researchers have focused on online hashing. We roughly divide the online hashing algorithms into unsupervised and supervised methods based on whether using the label as supervision information.

Unsupervised online hashing[21-23] learns hashing functions based on the intrinsic properties and regularity of the dataset. And the training process does not require supervised information, such as data labels. Sketching Hashing (SketchHash)[21] employ the frequent direction algorithm (FD)[24] to draw the sketch matrix of the data. SketchHash can obtain the key information of the samples, which solves the problem of excessive storage space occupation in large-scale image retrieval. To accelerate the rate of learning the hashing function, FasteR Online Sketching Hashing (FROSH)[22] utilizes the Subsampled Randomized Hadamard Transform (SRHT)[25] to compress the data, which further reduces the time complexity of SketchHash. Angular Quantization Online Hashing (AQOH)[23] preserves the original similarity relationship in Hamming space by minimizing the Hamming distance and cosine distance among the new and old datasets. As the unsupervised online hashing training does not involve the data label, it cannot preserve the original semantic similarity relationship. As a result, it has an inferior semantic ANN search performance.[26]

The supervised online hashing[13-17, 19, 20, 26-29] utilize the label information to supervise the process of learning hash functions. Furthermore, supervised online hashing[14-16, 26, 43] utilizes the Stochastic Gradient Descent mechanism(SGD)[30] to optimize the objective function, which can save storage space. Online Kernel Hashing(OKH)[13] uses the Passive-Aggressive (PA) [31] strategy to optimize the hash model while retaining the key information from both new and old datasets. Adapt Hashing (AdaptHash)[14] defines hinge-like loss as reducing the distance of the similar data pairs and increasing the dissimilar data pairs, which can improve the ANN search performance. Based on information theory, Online Hashing with Mutual Information(MIHash)[26] considers mutual information between neighbors and non-neighbors to reduce the ambiguities of neighborhoods in the Hamming space. Online Supervised Hashing (OSH)[15] proposes a two-step learning approach. It is achieved by assigning Error Correcting Output Codes(ECOC)[32-34] to the data and then learning the hash function to adapt the anonymous labeled data in the online learning framework. Hadamard Codebook-based Online Hashing (HCOH)[16] takes the Hadamard matrix as an ECOC and assigns the Hadamard codes to the data based on label information. As a result, HCOH is robust to noise. Online selection hashing (OSelH)[27] proposes an online bit selection method to capture data features, ensuring a low correlation between hash functions. Hadamard Matrix Guided Online Hashing (HMOH)[43] considers the Hadamard matrix columns as the target codes. HMOH adopts the hash function as a binary classifier to minimize the quantization errors in the relaxation process. Fast Class-wise Updating for Online Hashing(FCOH)[29] compensates for the poor adaption and inefficiency of online hashing by rapidly updating the hash function in terms of the data classes and optimizing the objective function by a semi-relaxation method.

While online hashing relies on traditional relaxation,[35, 36] the discrete space is converted to approximately continuous space, leading to inefficient optimization of the objective function. Online hashing[17, 19, 20] employs a discrete iterative scheme for optimizing the objective function to address the above problem. Balanced Similarity for Online Discrete Hashing (BSODH) designs two balancing factors that balance new and old datasets' similarity matrices to solve the data imbalance problem.[17] Scalable Supervised Online Hashing(SSOH) introduces an intermediate variable to replace the hash codes of the old dataset, which both alleviates the data imbalance problem and teaches the semantic information of the hash codes.[20] To balance the semantic similarity matrix between new and old datasets, Label Projection Online Hashing for Balanced Similarity (LPOH) maps labels into hash codes and utilizes the label projection matrix to attribute different weight values to labels.[19] However, optimizing the objective function is difficult as LPOH is affected by the binary code discretization and the unknown label projection matrix.

This paper proposes a novel supervised method to solve the data imbalance problem and maintain a similar relationship between new and old datasets. We term this

proposed method as a Label projection based on Hadamard codes for Online Hashing (LHOH). Figure 1 shows the LHOH consists of three major components: Hadamard codes, label projection, and similarity preservation. 1) First, we generate offline Hadamard matrices and then use Hadamard matric columns as Hadamard codes. Afterwards, as in (a) and (i), we assign Hadamard codes to the new and old data according to their label information. It is the first supervised learning process based on the label classes. 2) As in (b), (c), (f), and (h), this paper aligns the length of the Hadamard codes and the hash codes by employing the Locality Sensitive Hashing (LSH). We generate the label projection matrix V by projecting the labels into Hadamard codes. It is the second supervised learning process based on label information. Then, as in (d) and (g), we reconstruct the weight semantic similarity matrix by using the label projection matrix as the label weight values, which can avoid data imbalance. It is the third supervised learning process based on the label classes. 3) We set the semantic similarity preservation and codes error constraint to ensure the ANN search performance in the Hamming space. Of these, the semantic similarity preservation constraint ensures similarity relations between Hamming and original semantic space, which minimizes the error between the inner product of the hash codes and the semantic similarity matrix, as in (d), (e), and (g). As in (j) and (k), code error constraint preserves the characteristics of the data itself by minimizing the error between the hash code and the hash function of the new dataset.

As described above, the contributions of the proposed LHOH method are as follows.

- LHOH utilizes the Hadamard codes instead of hash codes as the label projection target domain. Furthermore, we consider the label projection matrix as label weight values, which can solve the data imbalance problem by balancing the similarity matrix between the new and old datasets. In addition, the label projection matrix can get a closed solution using the least square regression method, which facilitates the discrete iterative optimization of the objective function.
- To solve the data imbalance problem, LHOH unites label projection and similarity preservation mechanisms. That maintains the Hamming and semantic space similarity.
- Relationship and automatically balances the similarity matrix between the new and old datasets.
- To enhance the distinguishability of different labels' hash codes, LHOH sequentially implements the triple supervision processes based on labels, which includes the Hadamard codes, label projection, and learning similarity matrix. As a result, this can increase the classification ability of hash codes.

3. The Proposed Method

3.1. Problem Definition

Figure 2 shows the training and ANN search process based on online hashing. As new data streaming arrives, the online hashing will update the current hash function F(X) (a) and generate the hash code (b) in real time. At the same time, we can update the binary codes of the old dataset according to the hash function F(X), as shown in (c) and (d). Finally, when retrieving the nearest neighbor, the query dataset must generate binary codes depending on the hash function F(X). Online hashing computes the Hamming distance relationship between hash codes to query nearest neighbor samples in the new and old datasets, as shown in (e) and (f).

To ensure that the online hashing can achieve better nearest neighbor retrieval performance in Hamming space, the Hamming distance relationship between the new and old datasets should be consistent with their semantic relationships. During the training process, online hashing often establishes a semantic similarity matrix based on the label classes of the new and old datasets. However, there are significant differences in the number of samples and the label classes, which makes the established semantic similarity matrix sparse and asymmetric.

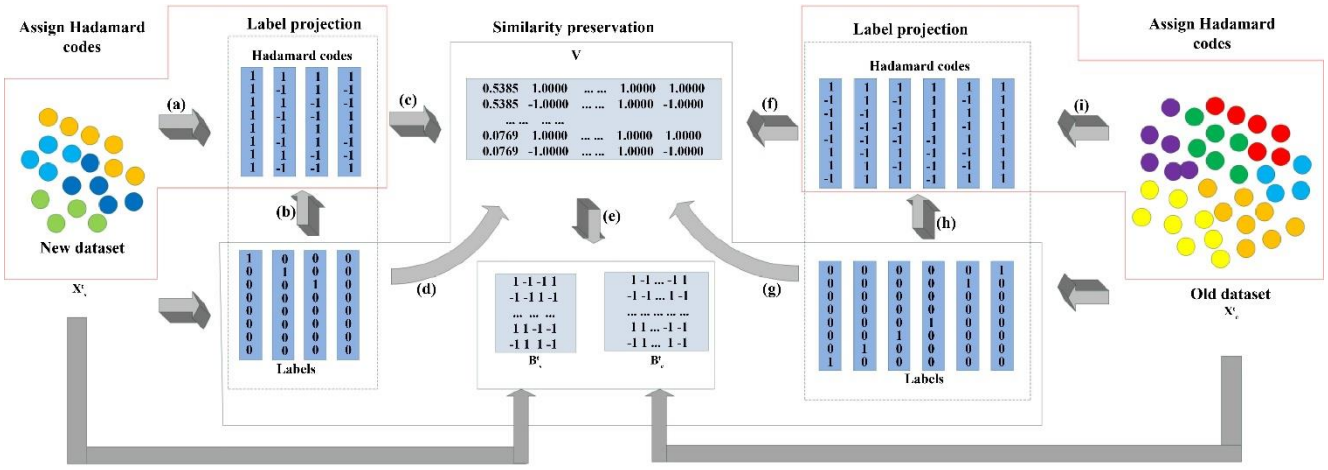

Fig. 1 Framework diagram of the LHOH algorithm process. We can divide the framework diagram into three significant parts: Hadamard codes, label projection, and similarity preservation. Assign new and old dataset Hadamard codes as parts (a) and (i). In parts (b) and (h), the labels project to Hadamard codes for obtaining the projection matrix V for both new and old datasets. In similarity preservation, parts (d), (e), and (g) demonstrate the preservation of similarity between Hamming and semantic space; (j) and (k) illustrate the preservation of the characteristics for the new data streaming itself.

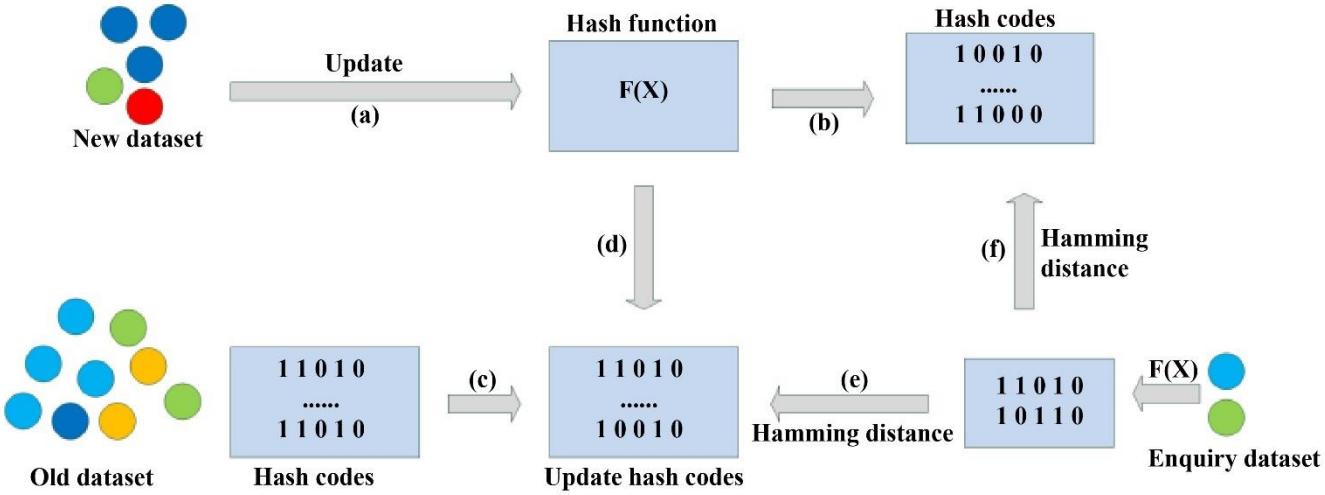

Fig. 2 Diagram of online hashing and nearest neighbor retrieval for online learning. Online hashing updates the hash function based on the new data streaming and generates binary codes for new and old datasets.

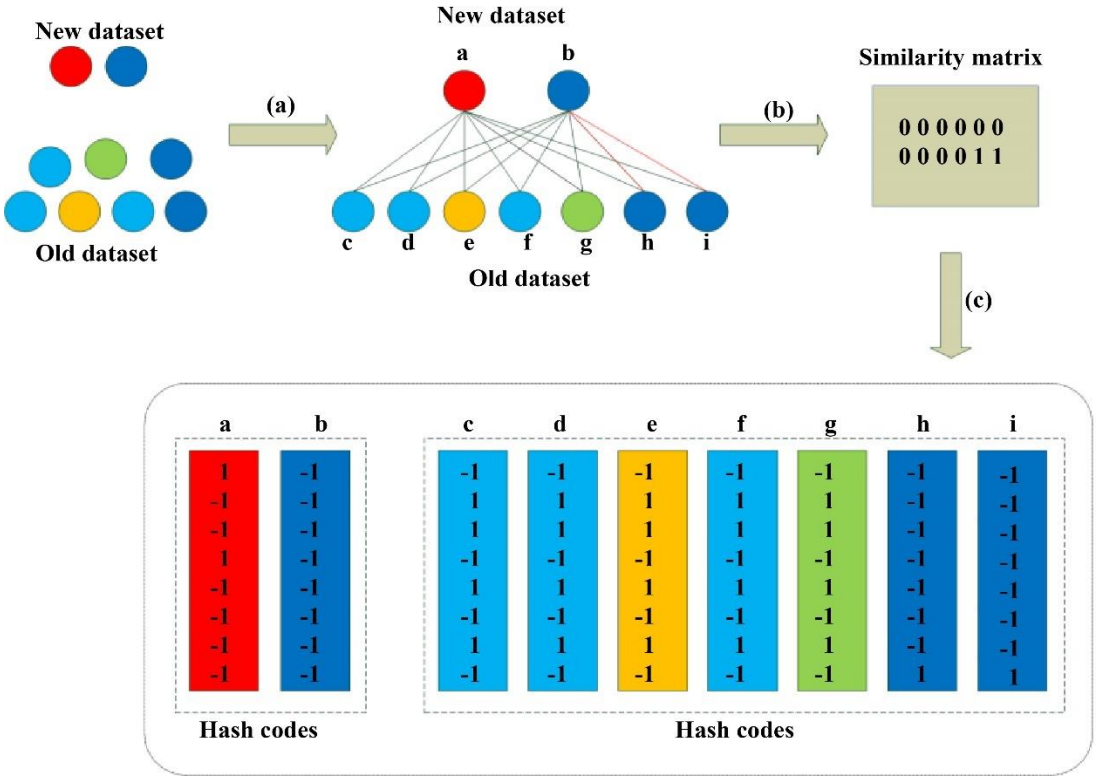

Fig. 3 Diagram of the data imbalance problem. Different colors of data mean that they have other labels.

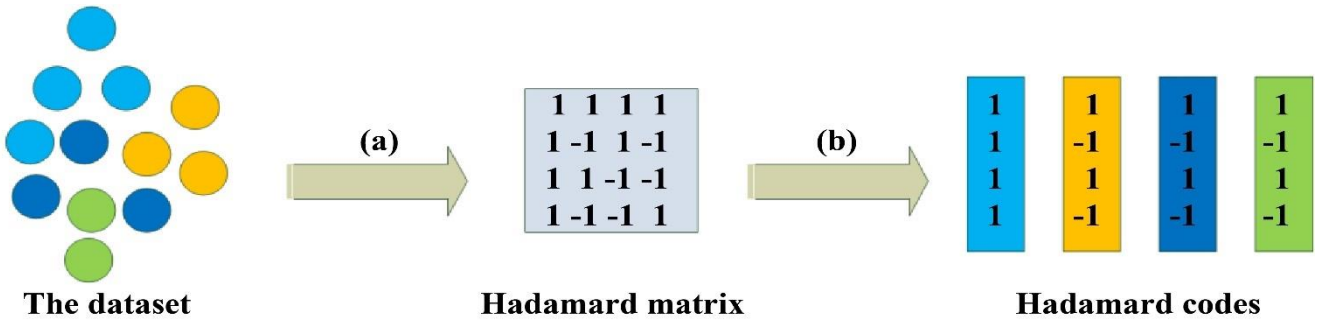

Fig. 4 Diagram of the process of distributing Hadamard codes. Hadamard codes are colored to match the corresponding data colors.

The number of dissimilar pairs is more extensive than similar pairs, which leads to a data imbalance problem. When the label of the new dataset is different from the old dataset. The new dataset and all the old datasets constitute other sample pairs. It results from the old dataset having similar binary codes, thus reducing the distinguishability between the old datasets' binary codes and weakening ANN search performance within the Hamming space.

Figure 3 illustrates the data imbalance problem, which minimizes the distinguishability of the binary codes. In Figure 3, we built the semantic similarity matrix (b) according to the similarity relationship between the labels of the new and old datasets (a). The similarity matrix has a value of 1 if the samples' labels are similar. Otherwise, the value is 0. There is a disparity between the new and old datasets, such as numbers and labels. Since the quantity of 0 is greater than 1, the similarity matrix is unbalanced and asymmetric, resulting in a data imbalance problem. For example, the labels are different between the new data a and the old data e, f, and g. So, the values in the similarity matrix are all 0. Assuming that the binary code of a is [1,-1,-1,1,-1,-1,-1,-1,-1], we maximize the Hamming distance between the new data a and the old data e, f, and g to preserve the original semantic similarity relationship between the samples in Hamming space. Then it will make the old data e, f, and g have the same binary codes [-1,1,1,-1,1,-1,1,-1], as shown in (c). Although the old data e, f, and g have different labels, they are less distinguishable from any sample with the same Hamming distance.

In this paper, to simulate the real-time training process of online hashing, the entire samples dataset X is divided into old and new datasets: $X=[X_s^t,X_e^t]\in R^{d\times n}$, where d denotes the dimensionality of the data features and n is the number of total samples. $X_s^t=[x_{s1}^t,\ldots\ldots,x_{sn_t}^t]\in R^{d\times n_t}$ denotes the feature vectors of the new dataset generated by t-batch, also called streaming data. And $L_s^t=[l_{s1}^t,\ldots\ldots,l_{sn_t}^t]\in\{1,0\}^{c\times n_t}$ is its label vector, where n_t is the size of the t-batch. The vector x_{si}^t in X_s^t means that the label of the i-th instance is l_{si}^t. $H_s^t=[h_{s1}^t,\ldots\ldots,h_{sn_t}^t]\in\{-1,1\}^{m\times n_t}$ denotes Hadamard codes matrix consistent with X_s^t. $X_e^t=[x_{e1}^t,\ldots\ldots,x_{e(n-n_t)}^t]\in R^{d\times(n-n_t)}$ denotes the feature vectors of the old dataset and $L_e^t=[l_{e1}^t,\ldots\ldots,l_{e(n-n_t)}^t]\in\{1,0\}^{c\times(n-n_t)}$ denotes its label vector. The corresponding Hadamard codes matrix of the old dataset is $H_e^t=[h_{e1}^t,\ldots\ldots,h_{e(n-n_t)}^t]\in\{-1,1\}^{m\times(n-n_t)}$. $L=[L_s^t,L_e^t]\in\{1,0\}^{c\times n}$ is the entire sample dataset label, where c is the number of classes of the sample. $H=[H_s^t,H_e^t]\in\{-1,1\}^{m\times n}$ denotes the corresponding Hadamard codes matrix of the dataset, where m is the length of the Hadamard codes matrix.

The hash codes for the new and old datasets are $B=[B_s^t,B_e^t]\in\{-1,1\}^{k\times n}$, where k denotes the length of the hash code. $B_s^t=[b_{s1}^t,\ldots\ldots,b_{sn_t}^t]\in\{-1,1\}^{k\times n_t}$ denotes the hash codes of the new dataset X_s^t. $B_e^t=[b_{e1}^t,\ldots\ldots,b_{e(n-n_t)}^t]\in\{-1,1\}^{k\times(n-n_t)}$ denotes the hash codes of the old dataset X_e^t. Alternatively, the hash function generated based on the t-batch dataset is defined as:

$$B=F(X)=sgn((W^t)^TX) \quad (1)$$

$W^t=[w_1^t,\ldots\ldots,w_k^t]\in R^{d\times k}$ is the projection matrix from the t-batch dataset to the binary hash codes. Sign function $sgn(x)$ returns +1 if the variable $x > 0$. Otherwise, it returns 1.

3.2. The Data Imbalance Problem

Online hashing aims to update the hash function only based on the new data streaming in real-time and preserve the original semantic similarity relationship between the new and old datasets in Hamming space. However, the number and classes are different between the new and old datasets, which leads to the data imbalance problem of the semantic similarity matrix. As a result, the ANN search performance of online hashing is inferior. This paper takes Hadamard codes as the projection target domain of the semantic labels, which simplifies the solution of the label projection matrix to a least square problem. Furthermore, we use the label projection matrix as the semantic label weight to automatically balance the semantic similarity matrix.

3.2.1. Hadamard codes

To avoid the data imbalance problem in the semantic similarity matrix, we utilize the label projection weight matrix A to assign different weight values to the labels L_s and L_e.[19] The semantic similarity preservation objective function is defined in Equation 2. Where B_s and B_e denote the hash codes of the new and old datasets, respectively. A is the projection matrix mapping the semantic labels to the hash codes.

$$\min_{B_s,B_e,A}\left\|(B_s)^TB_e-k(AL_s)^TAL_e\right\|_F^2 \quad (2)$$

In Equation 2, B_s and B_e are discrete binary codes. A is an unknown matrix, which makes it a challenge to optimize this objective function directly. This paper proposes a two-step mechanism to solve the above problem. The weight matrix A is solved first. Then the objective function is optimized.

The projection matrix $V\in R^{c\times k}$ substitutes the label projection weight matrix A. We can pre-calculate the label projection matrix V, mapping the data label L to the Hadamard code H and using it as the label weight value. The projection matrix V minimizes the error between the projection target domain H and the projection value L. The function is defined in Equation 3.

$$\min_V\left\|H^T-L^TV\right\|_F^2 \quad (3)$$

To obtain the closed solution of the label projection matrix, we replace the hash code B with the Hadamard code H to be the target domain for the projection of the label L in the training dataset. The convergence of hash codes generated by the same label data, and the distinguishability generated by dissimilar is the goal of label projection to hash

codes. [2] In this paper, Hadamard codes pre-meet the projection target domain expectation. The same label data have consistent Hadamard codes, and dissimilar have distinguishable Hadamard codes. Therefore, Hadamard codes can be ideal target codes for label projection. In addition, Hadamard codes have the following advantages over hash codes. 1) We can obtain Hadamard codes from the Hadamard matrix columns.[37] It only has two values, including +1 and -1, which conform to binary codes' characteristics; 2) Hadamard codes satisfy orthogonality and equilibrium. Any code pairs have maximum and equal Hamming distance, which provides more substantial error correction.

In this paper, the length of the Hadamard code H is $m=2^a$(a=1,2,3,4…….) and the number of the label is c. To ensure that each label has a different Hadamard code H, the relationship between m and c should satisfy $m\geq c$. In addition, the Hadamard code length m should be larger than the hash code length k, defined as $m\geq k$. To satisfy all three conditions simultaneously, we define the process of selecting the Hadamard code length as in Equation 4.

$$m=\min\{l|l=2^a, l\geq k, l\geq c, a=1,2,3,4\ldots\ldots\} \quad (4)$$

Inspired by HCOH and HMOH, Figure 4 shows the process of assigning Hadamard codes to the dataset, which consists of two main steps. 1) Pre-construct a Hadamard matrix of length m (a) and use the Hadamard matrix columns as Hadamard codes; 2) we assign Hadamard codes according to labels. This achieves the first supervised learning process based on label information. The distribution principle follows assigning the same Hadamard codes to the same label data and the difference to the dissimilar, as shown in Figure 4(b).

3.2.2. Label Projection Matrix

Following section 3.2.1, Hadamard codes are assigned to the data according to the label, generating the Hadamard codes matrix H_s and H_e for the new and old datasets. In addition, we convert the solution of the label projection matrix V into a least square problem, which implements the second supervised learning process based on label classes. Online hashing updates the hash function according to new streaming in real-time. When new streaming arrives in t-batch, the new and old dataset label vectors L_s^t and L_e^t project onto their Hadamard codes H_s^t and H_s^t. The formula is defined as follows：

$$\min_{V^t} \left\|(H_s^t)^T-(L_s^t)^T V^t\right\|_F^2+\left\|(H_e^t)^T-(L_e^t)^T V^t\right\|_F^2+\mu^t\|V^t\|_F^2 \quad (5)$$

$V^t\in R^{c\times k}$ is the projection matrix which maps the label vectors of the new and old datasets onto the corresponding Hadamard codes matrix in t-batch. μ^t is the prevent overfitting coefficients in t-batch.

This paper considers Hadamard codes instead of hash codes as the target domain for label projection. The Hadamard code length m should be the same as the hash code k. The pre-designed Hadamard code length may be longer than the hash code length, as in section 3.2.1. So it is necessary to align the Hadamard codes with hash codes. Locality Sensitive Hashing (LSH)[38] can reduce the Hadamard code length while preserving their similar structure.[16, 43] Therefore, we use the LSH to align m with k. The formula is defined as follows：

$$\widetilde{H}=sgn(\widetilde{W}^T H) \quad (6)$$

When $m>k$, $\widetilde{W}^T\in R^{m\times k}$ is the random Gaussian projection matrix. $m=k$, $\widetilde{W}^T\in R^{m\times k}$ is the identity matrix. H is the Hadamard code assigned to the data, and $\widetilde{H}$ denotes Hadamard code for the aligned hash code length.

Combining Equations 5 and 6, the Equation for the projection of labels to Hadamard codes is redefined as follows:

$$\min_{V^t} \left\|(sgn(\widetilde{W}^T H_s^t))^T-(L_s^t)^T V^t\right\|_F^2+\left\|(sgn(\widetilde{W}^T H_e^t))^T-(L_e^t)^T V^t\right\|_F^2+\mu^t\|V^t\|_F^2 \quad (7)$$

Let $\widetilde{H}_s^t=sgn(\widetilde{W}^T H_s^t)$, $\widetilde{H}_e^t=sgn(\widetilde{W}^T H_e^t)$. Equation 7 is rewritten as follows:

$$\min_{V^t} \left\|(\widetilde{H}_s^t)^T-(L_s^t)^T V^t\right\|_F^2+\left\|(\widetilde{H}_e^t)^T-(L_e^t)^T V^t\right\|_F^2+\mu^t\|V^t\|_F^2 \quad (8)$$

In contrast to LPOH, we project the labels to Hadamard codes and consider the label projection matrix as the label weight value. It can solve the data imbalance problem by balancing the semantic similarity matrix between the new and old datasets. This paper constructs semantic similarity matrices for new and old datasets by embedding labels to achieve the third supervised learning process based on label classes. It helps the hash codes generated from the same labels data to converge and the differences to be inconsistent. When the t-batch of new data streaming arrives, we can attribute weight values to the data labels using the label projection matrix V. $(V^t)^T L_s^t$ and $(V^t)^T L_e^t$ denote the weight labels of the new and the old datasets. The balanced semantic similarity matrix $S^{'t}$ between the new and old datasets is defined as follows:

$$S^{'t}=((V^t)^T L_s^t)^T((V^t)^T L_e^t) \quad (9)$$

$V^t\in R^{c\times k}$ is the projection matrix which maps the label vectors of the new and old datasets onto the corresponding Hadamard codes matrix in t-batch.

Let $\widetilde{L}_s^t=(V^t)^T L_s^t$, $\widetilde{L}_e^t=(V^t)^T L_e^t$. Equation 9 can be rewritten as follows:

$$S^{'t}=(\widetilde{L}_s^t)^T\widetilde{L}_e^t \quad (10)$$

3.3. Similarity Preservation

To ensure the semantic similarity of Hamming space is consistent with the original, we try to minimize the inner product of hash codes and the semantic similarity matrix

between the new and old datasets. As mentioned in Section 3.1, the direct construction of the similarity matrix suffers from data imbalance. In Section 3.2, this paper projects the label L onto the Hadamard codes matrix H and utilizes the projection weight matrix V to balance its similarity matrix. Combining Equation 10, preserving the consistent similarity within the Hamming and original space is defined as follows:

$$\min_{B_s^t,B_e^t} \left\|(B_s^t)^T B_e^t - kS^{'t}\right\|_F^2 \quad (11)$$

Where $B_s^t \in \{-1,1\}^{k\times n_t}$ is the hash codes obtained from the new dataset learning in *t*-batch, $B_e^t \in \{-1,1\}^{k\times(n-n_t)}$ is the hash codes obtained from the old dataset learning in *t*-batch, and $S^{'t} \in \{1,-1\}^{n_t\times(n-n_t)}$ is the balanced semantic similarity matrix of labels for the new and old datasets in *t*-batch. From Equation 11, the old dataset participates in the updating process of the hashing function. That ensures the generated online hash model can achieve a satisfied ANN search performance for both new and old datasets.

To learn the structure of the new data features, we need to quantify the error of the hashing function and hash codes for new streaming data. According to Equation 1, the quantization between the hashing function and hash codes is defined as follows:

$$\min_{B_s^t,W^t} \left\|B_s^t - sgn((W^t)^T X_s^t)\right\|_F^2 + \lambda^t \|W^t\|_F^2 \quad (12)$$

λ^t is the coefficient to avoid the overfitting problem in the *t*-batch.

Since *sgn*(.) is a discrete notation, optimizing Equation 12 is an NP-hard problem. We consider a linear function $(W^t)^T X_s^t$ instead of the discrete function $sgn((W^t)^T X_s^t)$, which transforms the discrete problem into a linear regression problem. Equation 12 is rewritten as follows:

$$\min_{B_s^t,W^t} \left\|B_s^t - (W^t)^T X_s^t\right\|_F^2 + \lambda^t \|W^t\|_F^2 \quad (13)$$

According to Equations 8, 11, and 13, the overall objective function is defined as follows:

$$\min_{B_s^t,B_e^t,W^t,V^t} \left\|(\widetilde{H}_s^t)^T - (L_s^t)^T V^t\right\|_F^2 + \left\|(\widetilde{H}_e^t)^T - (L_e^t)^T V^t\right\|_F^2 + \left\|(B_s^t)^T B_e^t - kS^{'t}\right\|_F^2 + \varphi^t \left\|B_s^t - (W^t)^T X_s^t\right\|_F^2 + \mu^t \|V^t\|_F^2 + \lambda^t \|W^t\|_F^2 \quad (14)$$

Where φ^t is the balance coefficient, $S^{'t} = (\widetilde{L}_s^t)^T \widetilde{L}_e^t$.

3.4. Optimization

Due to the constraints of the binary codes in Equation 14, the objective function is non-convex which makes the objective variables hard to optimize. This paper updates the objective variables using an alternating iterative optimization algorithm. We can fix other variables when updating a variable until the objective function converges in each round. The optimization process is as follows:

3.4.1. Updating V^t

When updating V^t, we can consider $\widetilde{H}_s^t$, $\widetilde{H}_e^t$, L_e^t and L_s^t as a fixed variable, the objective function (14) is redefined as follows:

$$\min_{V^t} \left\|(\widetilde{H}_s^t)^T - (L_s^t)^T V^t\right\|^2 + \left\|(\widetilde{H}_e^t)^T - (L_e^t)^T V^t\right\|^2 + \mu^t \|V^t\|^2 \quad (15)$$

We directly solve the variable V^t using the least square regression method. The solution is defined as follows:

$$V^t = (L_s^t (L_s^t)^T + L_e^t (L_e^t)^T + \mu^t I_1)^{-1} (L_s^t (\widetilde{H}_s^t)^T + L_e^t (\widetilde{H}_e^t)^T) \quad (16)$$

Where $I_1 \in R^{c\times c}$ is an identity matrix.

3.4.2. Updating W^t

Fixing the variables B_s^t and B_e^t, the objective function (14) is redefined as follows:

$$\min_{W^t} \varphi^t \left\|B_s^t - (W^t)^T X_s^t\right\|_F^2 + \lambda^t \|W^t\|_F^2 \quad (17)$$

Therefore, the closed solution for W^t can be obtained as:

$$W^t = \varphi^t (\varphi^t X_s^t X_s^t)^T + \lambda^t I_2)^{-1} X_s^t (B_s^t)^T \quad (18)$$

Where $I_2 \in R^{d\times d}$ is an identity matrix.

3.4.3. Updating B_e^t

Fixing the variables W^t and B_s^t, the objective function 14 is redefined as follows:

$$\min_{B_e^t} \left\|(B_s^t)^T B_e^t - kS^{'t}\right\|_F^2 \quad (19)$$

The squared Frobenius norm replaces with the L_1 norm.[17, 39] Equation 19 can be rewritten as follow:

$$\min_{B_e^t} \left\|(B_s^t)^T B_e^t - kS^{'t}\right\|_1 \quad (20)$$

Solving for Equation 20 yields the closed solution of B_e^t as follows:

$$B_e^t = sgn(B_s^t S^{'t}) \quad (21)$$

3.4.4. Updating B_s^t

Fixing the variables W^t and B_e^t, the objective function 14 is redefined as follows:

$$\min_{B_s^t} \left\|(B_s^t)^T B_e^t - kS^{'t}\right\|_F^2 + \varphi^t \left\|B_s^t - (W^t)^T X_s^t\right\|_F^2 \quad (22)$$

By simplifying Equation 22 as the *tr*(.) operation of the matrix, Equation 22 can be rewritten as follow:

$$\min_{B_s^t} \left\|(B_e^t)^T B_s^t\right\|_F^2 + \left\|kS^{'t}\right\|_F^2 - 2tr(kS^{'t}(B_e^t)^T B_s^t) + \varphi^t(\|B_s^t\|_F^2 + \left\|(W^t)^T X_s^t\right\|_F^2 - 2tr((X_s^t)^T W^t B_s^t)) \quad (23)$$

Where $tr(.)$ is the trace norm. By removing the constant term unrelated to B_s^t, Equation 23 can be rewritten as follow

$$\min_{B_s^t} \left\| (B_e^t)^T B_s^t \right\|_F^2 - 2tr(C^T B_s^t) \quad (24)$$

Where $C=kB_e^t(S'^t)^T+\varphi^t(W^t)^T$.

Due to B_s^t is discrete, solving B_s^t is an NP-hard problem. Inspired by the Discrete Cycle Coordinate Descent Method (DCC) proposed in SDH,[1] this paper solves each line of the binary codes B_s^t by fixing the other lines. We describe the process as follows：

Firstly, let $\bar{c}_i^t$, $\bar{b}_{si}^t$ and $\bar{b}_{ei}^t$ be the i-th row of matrices C, B_s^t and B_e^t, respectively. $\bar{C}$, $\bar{B}_s^t$ and $\bar{B}_e^t$ are the matrices obtained by removing the i-th row from matrix C, B_s^t and B_e^t, respectively. Equation 24 is redefined as follows:

$$\min_{B_s^t} \left\| (\bar{B}_e^t)^T \bar{B}_s^t \right\|_F^2 + \left\| (\bar{b}_{ei}^t)^T \bar{b}_{si}^t \right\|_F^2 + 2tr((\bar{B}_s^t)^T \bar{B}_e^t (\bar{b}_{ei}^t)^T \bar{b}_{si}^t) - 2tr(\bar{C}^T \bar{B}_s^t) - 2tr((\bar{c}_i^t)^T \bar{b}_{si}^t)\, \bar{c}_i^t \quad (25)$$

Then, the constant term in Equation 25 is removed as follows:

$$\min_{\bar{b}_{si}^t} tr(((\bar{B}_s^t)^T \bar{B}_e^t (\bar{b}_{ei}^t)^T - (\bar{c}_i^t)^T) \bar{b}_{si}^t) \quad (26)$$

Finally, the value of each line in B_s^t is computed as follows:

$$\bar{b}_{si}^t = sgn(\bar{c}_i^t - \bar{b}_{ei}^t (\bar{B}_e^t)^T \bar{B}_s^t) \quad (27)$$

This paper uses an iterative alternating update method, which updates the variables V^t, W^t, B_e^t and B_s^t sequentially until the objective function converges.

Algorithm 1 summarizes the main procedures of the proposed BSODH.

Algorithm 1: Label projection based on Hadamard codes for Online Hashing(LHOH)

Input：

Training dataset: $X=[X_s^t, X_e^t] \in R^{d \times n}$

Training dataset labels: $L=[L_s^t, L_e^t] \in \{1,0\}^{c \times n}$

Hadamard codes matrix corresponding to training dataset: $H=[H_s^t, H_e^t] \in \{-1,1\}^{m \times n}$

Length of the hash codes: k

Parameters: φ^t, μ^t, λ^t

(1) Set the total number of streaming data batches T;
(2) According to Equation 4, the value of the Hadamard codes matrix length is determined, and the Hadamard codes matrix is constructed offline;
(3) **if** $m=k$ **then**
(4) $\widetilde{W}^T$ is an identity matrix;
(5) **else**
(6) $\widetilde{W}^T$ is a random Gaussian projection matrix;
(7) According to Equation 6, Hadamard codes are aligned with hash codes.
(8) **for** $t=1 \rightarrow T$ **do**
(9) Define the new data streaming generated in t-batch as X_s^t.
(10) **if** $t=1$, **then**
(11) Initialize W^t, B_s^t, B_e^t;
(12) **else**
(13) According to Equation 16, update V^t;
(14) According to Equation 10, compute S'^t;
(15) Initialize B_s^t;
(16) According to Equation 18, update W^t;
(17) According to Equation 21, update B_e^t;
(18) **repeat**
(19) **if** $i=1 \rightarrow k$ **then**
(20) According to Equation 27, update $\bar{b}_{si}^t$;
(21) **end for**
(22) **until**(Convergence or reaching the maximum number of iterations)
(23) **end if**
(24)**end for**
(25)Update X and B;
(26)Set $W=W^t$;
(27)Compute $B=sgn(W^T X)$;

Output: W, B

4. Results

4.1. Datasets

This paper adopts three widely used public datasets, including CIFAR-10, MNIST, and Places205.

The CIFAR-10 dataset consists of 60,000 samples containing ten classes, and each sample is 4096 dimensional CNN vectors.[40] Based on the paper,[17] we divide the whole dataset into two parts: 59000 data as a retrieval set and 10000 data as a test set. To suit the need for online hashing, we divide the entire retrieval set into new and old datasets. Among them, the new dataset contains 20,000 data. The new dataset arrives in batches of 2000 data each, divided into ten batches.

The MNIST datatset[41] consists of 70,000 handwritten digital image data, where each data dimension is 784 and contains ten classes. Among them, we extract 100 data from each class to form the test set, and the remaining data are the retrieval set. The entire retrieval set includes old and new datasets. The new dataset contains 20,000 samples. The new dataset arrives in batches of 2000 data each, divided into ten batches.

Places205 has 250 million images containing 205 classes. We extract the image features by the AlexNet fc7 layer[42] and then downscale to 128 dimensions by PCA. In this experiment, the entire Places205 extract datasets of 16 classes with a total of 204,715 images. Each class extracts 100 images as the test set and the rest as the retrieval set. The entire retrieval set contains old and new datasets. Due to the large size of the Places205 dataset, we extract 40,000 data as new datasets for the retrieval set. The new dataset arrives in batches of 4000 data each, divided into ten batches.

Table 1. The *m*AP and *m*AP@1000 results of various methods on CIFAR-10 with different code lengths.

Method	*m*AP					*m*AP@1000				
	16-bit	32-bit	48-bit	64-bit	128-bit	16-bit	32-bit	48-bit	64-bit	128-bit
OKH	0.1002	0.1664	0.2981	0.3301	0.3557	0.1063	0.2361	0.4315	0.4736	0.5164
AdaptHash	0.2546	0.2155	0.1964	0.1983	0.2191	0.3205	0.3527	0.3917	0.3811	0.3177
SketchHash	0.2932	0.3136	0.3373	-	-	0.4129	0.4714	0.4984	-	-
OSH	0.1242	0.1163	0.1221	0.1306	0.1349	0.1650	0.1596	0.1561	0.1412	0.1943
MIHash	0.6382	0.6301	0.6141	0.6009	0.5634	0.6575	0.6105	0.6827	0.6595	0.6248
BSODH	0.6402	0.6899	0.6748	0.6926	0.6992	0.7103	0.7327	0.7370	0.7403	0.7562
LHOH	**0.6688**	**0.7025**	**0.6790**	**0.7185**	**0.7299**	**0.7208**	**0.7554**	**0.7494**	**0.7679**	**0.7811**

Table 2. The *m*AP and *m*AP@1000 results of various methods on MNIST with different code lengths.

Method	*m*AP					*m*AP@1000				
	16-bit	32-bit	48-bit	64-bit	128-bit	16-bit	32-bit	48-bit	64-bit	128-bit
OKH	0.1002	0.1802	0.3023	0.3334	0.3983	0.1103	0.4598	0.6025	0.6275	0.7229
AdaptHash	0.1938	0.2235	0.2047	0.2589	0.2228	0.3187	0.2513	0.3454	0.3143	0.3747
SketchHash	0.3331	0.3482	0.3766	-	-	0.5696	0.6774	0.7187	-	-
OSH	0.1431	0.1330	0.1287	0.1633	0.1671	0.1527	0.1832	0.1892	0.2213	0.2760
MIHash	0.6300	0.7099	0.6884	0.7333	0.7241	0.7579	0.8030	0.7970	0.8348	0.8450
BSODH	0.6589	0.7098	0.7229	0.7386	0.7392	0.7614	0.7837	0.8167	0.8297	0.8391
LHOH	**0.6716**	**0.7158**	**0.7252**	**0.7548**	**0.7610**	**0.7638**	**0.8180**	**0.8201**	**0.8457**	**0.8614**

Table 3. The *m*AP and *m*AP@1000 results of various methods on Places205 with different code lengths.

Method	*m*AP					*m*AP@1000				
	16-bit	32-bit	48-bit	64-bit	128-bit	16-bit	32-bit	48-bit	64-bit	128-bit
OKH	0.0625	0.1353	0.3382	0.3962	0.4364	0.0679	0.3570	0.5628	0.6089	0.6590
AdaptHash	0.1731	0.2348	0.1484	0.1597	0.1851	0.3157	0.3711	0.2031	0.2552	0.2590
SketchHash	0.3303	0.3791	0.4007	-	-	0.5352	0.6164	0.6511	-	-
OSH	0.0914	0.0910	0.1064	0.1018	0.1118	0.1286	0.1684	0.1572	0.1849	03397
MIHash	0.5910	0.6259	0.6489	0.6933	0.6454	0.6821	0.7143	0.7240	0.7294	0.7370
BSODH	0.6226	0.6610	**0.6787**	0.6933	0.7096	0.7011	0.7332	0.7422	0.7631	0.7724
LHOH	**0.6336**	**0.6874**	0.6577	**0.7054**	**0.7167**	**0.7068**	**0.7624**	**0.7498**	**0.7680**	**0.7855**

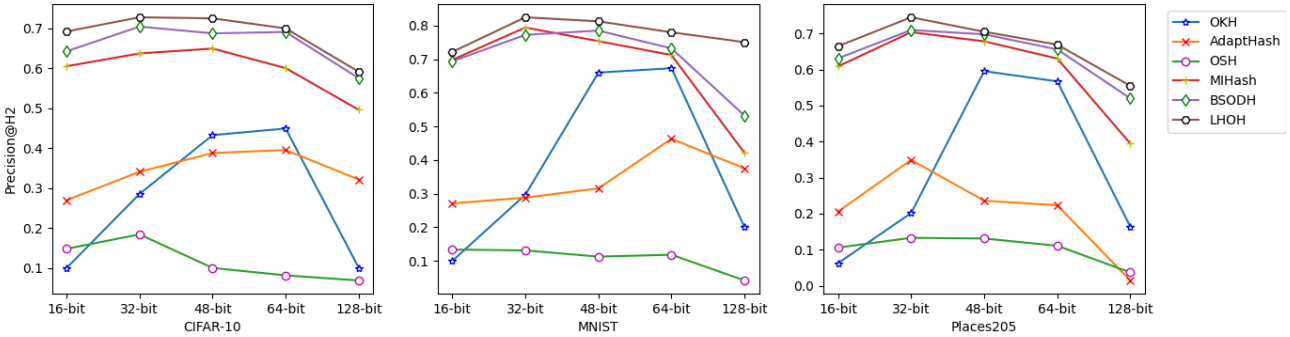

Fig. 5 The Precision@H2 results of various methods on three datasets with different code lengths.

4.2. Baselines and Evaluated Metrics

We compare the proposed method with six more advanced online hashing methods, including OKH, AdaptHash, SketchHash, OSH, MIHash, and BSODH. LHOH. In addition, the baselines are on the same datasets. We adopt the four metrics to measure the performance of nearest neighbor retrieval for each method.

The four metrics include the mean Average Precision(*m*AP), *m*AP on the top-1,000 retrieved items(*m*AP@1000), precision within a Hamming ball of radius two centered on each query(Precision@H2), and mean precision of the top-R retrieved neighbors (Precision@K).

4.3. Overall Comparison with Baselines

LHOH and the baselines map the samples to the binary codes with 16-bit, 32-bit, 48-bit, 64-bit, and 128-bit lengths, respectively. They retrieve the nearest neighbor samples based on Hamming distance.

4.3.1. *m*AP and *m*AP@1000

Table 1 shows the *m*AP and *m*AP@1000 results for LHOH and the baselines on the CIFAR-10 datasets. Where bold and underlining indicate the best and second-best results, respectively. From Table 1, we can draw that the proposed LHOH has the best ANN search performance. When the hash code length is 16-bit, 32-bit, 48-bit, 64-bit, and 128-bit, respectively, LHOH improves 2.86%, 1.26%,

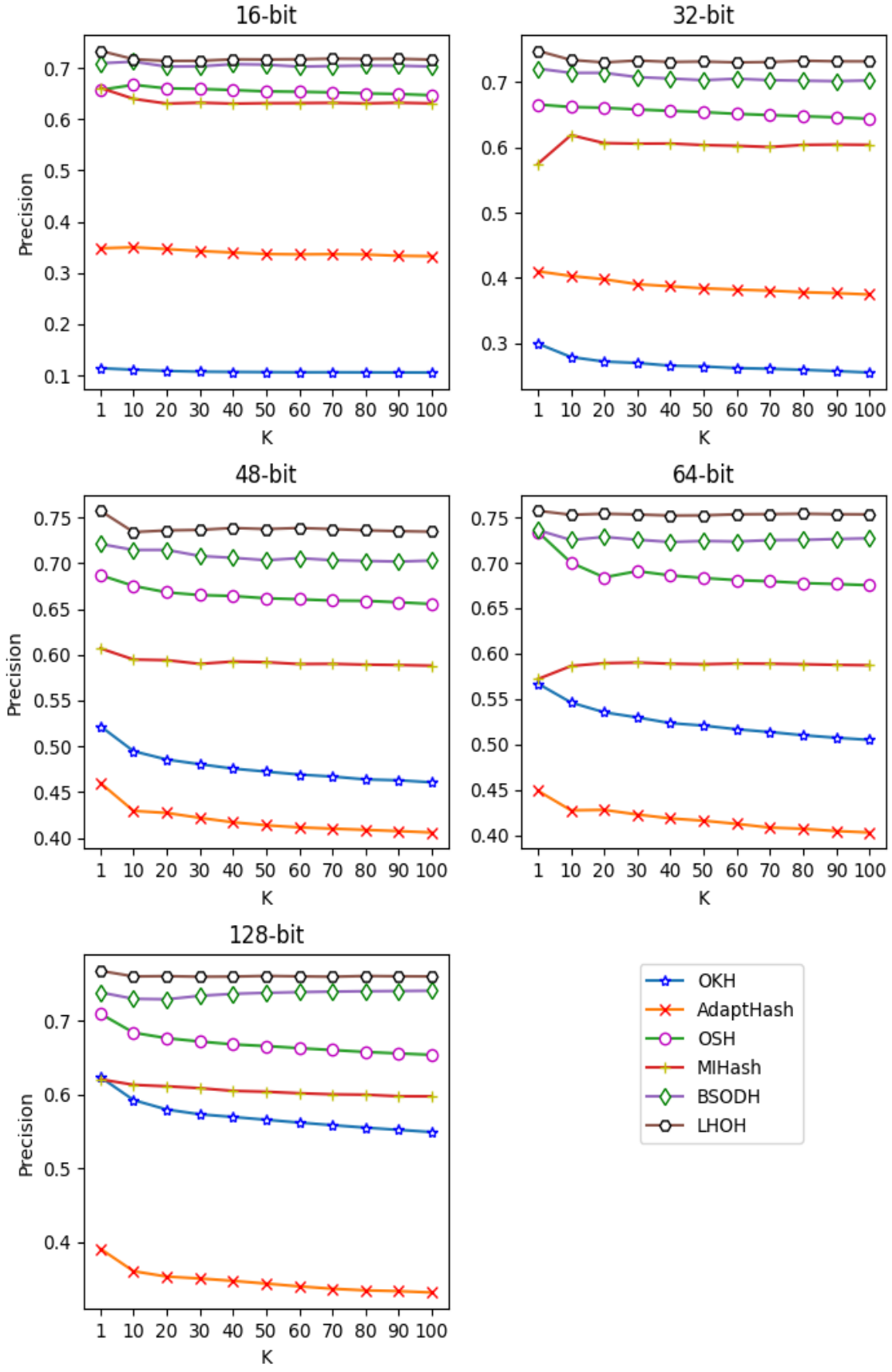

Fig. 6 The Precision@K results of various methods on CIFAR-10 with different code lengths.

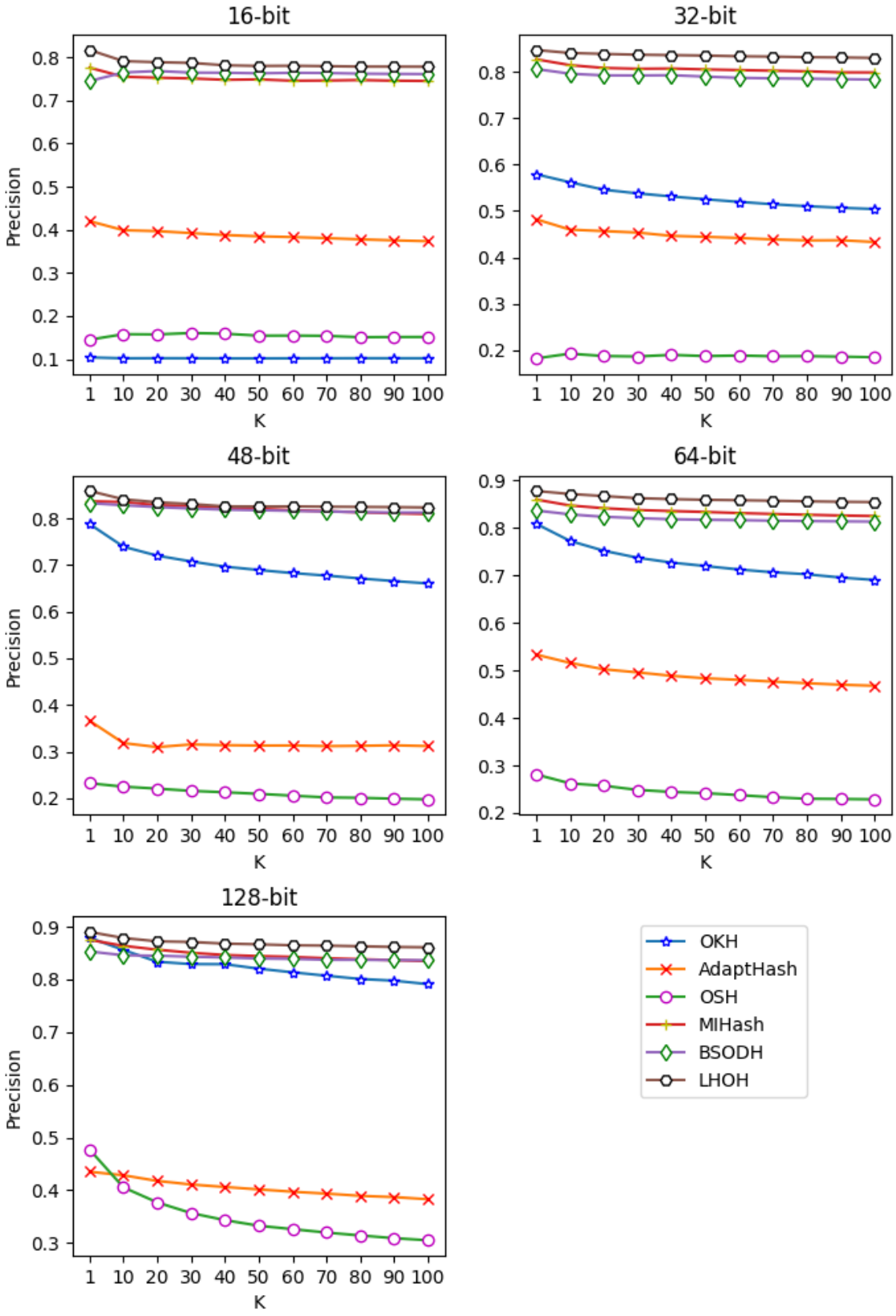

Fig. 7 The Precision@K results of various methods on MNIST with different code lengths.

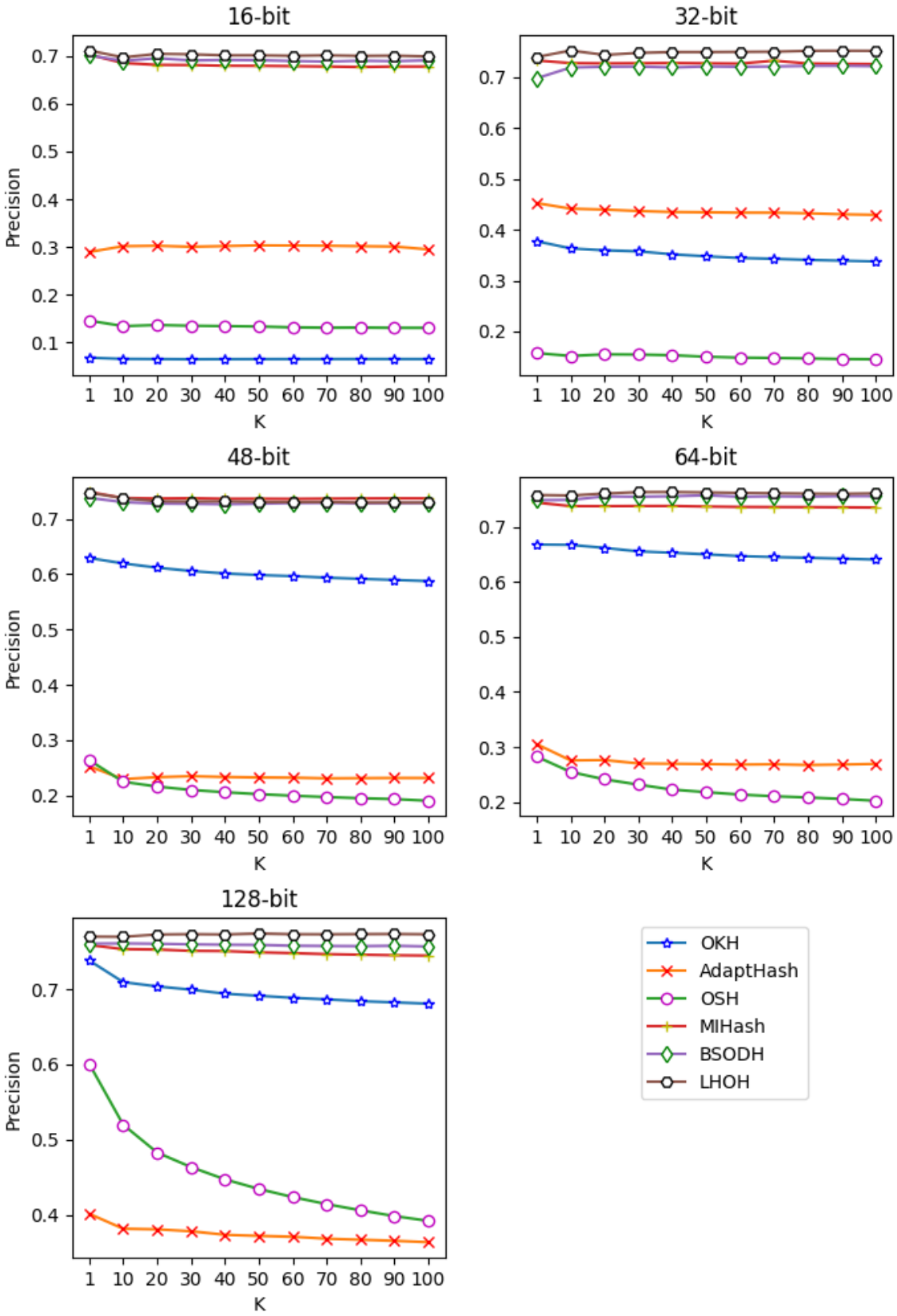

Fig. 8 The Precision@K results of various methods on Places205 with different code lengths.

0.42%, 2.59%, and 3.07% over the second-best baseline BSODH in terms of *m*AP evaluation metric. On the *m*AP@1000 evaluation metric, LHOH improves over the second-best baselines BSODH by 1.05%, 2.27%, 1.24%, 2.76%, and 2.49%, respectively.

In addition, Tables 2 and 3 show the *m*AP and *m*AP@1000 results with different length hash codes on the MNIST and Places205 datasets, respectively. Tables 2 and 3 show that LHOH has a superior ANN search performance than the other baselines. When the hash code length is 128-bit, compared to the second-best baseline, the proposed LHOH achieves improvements of 2.18% and 1.64% on the MNIST, respectively. When the hash code length is 32-bit, the proposed LHOH improves by 2.64% and 2.92% on the Places205 dataset.

Table 3 shows that the proposed LHOH performs slightly worse than BSODH with 48-bit binary code. Because LSH causes the length loss of Hadamard codes, affecting the ability to describe samples with smaller feature vector dimensions. That makes it challenging for different samples.

4.3.2. Precision@H2

Figure 5 shows the Precision@H2 metrics on three datasets, CIFAR-10, MNIST, and Places205. In Figure 5, the Precision@H2 metric of LHOH outperforms the other baseline methods on all three datasets. The figure shows that the Precision@H2 of all methods decreases when the hash code length reaches 128-bit. The possible reason is that the precision search space becomes too large to learn with good accuracy performance as the hash code grows.

The Precision@H2 retrieval precision results of all experimental methods fluctuate with the change in the hash code length. On the MNIST, OKH fluctuates the most, and the proposed LHOH fluctuates the least with the best experimental performance.

4.3.3. Precision@K

Figures 6, 7, and 8 show the Precision@K versus the baseline methods on the CIFAR-10, MNIST, and Places205 datasets, respectively. Spanning ten neighbors simultaneously, we set the neighbor search range to 1-100 in this experiment. That means we can get 11 experimental results per bit in each method. Figures 6, 7, and 8 show that the proposed LHOH is higher than other baseline methods on different datasets. The experimental data fluctuate less with a more stable retrieval performance.

When the hash code length is greater than or equal to 32-bit, Figure 6 shows that LHOH performs much better than the other baseline methods on the CIFAR-10 dataset. In Figures 7 and 8, when the hash code length becomes 48-bit, the ANN search performance of LHOH is more negligible than the baseline methods. LSH needs to be used to align Hadamard codes with the hash codes. When the hash codes length is 48-bit, offline generated Hadamard codes length is 64-bit. That results in the loss of the Hadamard code length.

However, LHOH performs better than other methods, such as BSODH and MIHash.

Figures 6, 7, and 8 show the Precision@K performance on the three different datasets. The ANN search performance improvement of LHOH is most remarkable in the CIFAR-10 dataset and less in the Places205 dataset. CIFAR-10 and MNIST have high and low dimensional data in this experiment, which affects the samples' expressiveness. Further, it results from achieving different ANN search performances on CIFAR-10 and MNIST.

In summary, we select whatever the value of the hash codes length or the datasets of CIFAR-10, MNIST, and Places205. The proposed LHOH is valid by its excellent performance in mAP, mAP@1000, Precision@H2, and Precision@K evaluation metrics.

4.4. Parameter sensitivity analysis

In this section, we discuss the influence of the parameters μ^t, φ^t, and λ^t on the ANN search performance. We set the ANN search experiments on the CIFAR-10, MINIST, and Places205 datasets. Simultaneously, we use *m*AP as the performance evaluation criterion with a 64-bit length and set the value range of all parameters μ^t, φ^t, and λ^t to [0.1,1].

4.4.1. Influence of μ^t

μt can prevent overfitting by adjusting the corresponding label projection matrix "V" . Figures 9 (a), (b), and (c) show the mAP performances on CIFAR-10, MINIST, and Places205 with different μt values, respectively. From Figure 9, we choose the ideal parameter μ^t=[0.3, 0.3, 0.4] for the CIFAR-10, MINIST, and Places205 datasets

4.4.2. Influence of φ^t

φ^t is the balance parameter that adjusts the error between the hash code generated by the new dataset and its hashing function. Figures 10 (d), (e), and (f) show the *m*AP results of parameter φ^t on three datasets, CIFAR-10, MINIST, and Places205, respectively. From Figure 10, we can set parameter φ^t =[0.6, 0.3, 0.8] for CIFAR-10, MINIST, and Places205 datasets.

4.4.3. Influence of λ^t

λ^t adjusts the projection matrix W to prevent overfitting. Figure 11 (g), (h), and (i) show the *m*AP results of parameter λ^t on the three experimental datasets CIFAR-10, MINIST, and Places205, respectively. As demonstrated in Figure 11, the most desirable developments in the above three datasets are for parameters λ^t =[0.2,0.5,0.7].

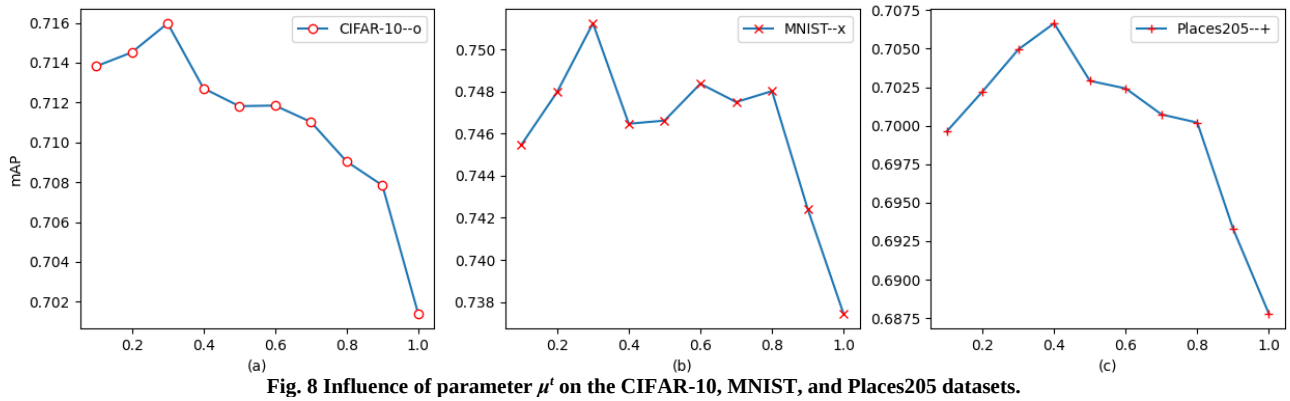

Fig. 8 Influence of parameter μ^t on the CIFAR-10, MNIST, and Places205 datasets.

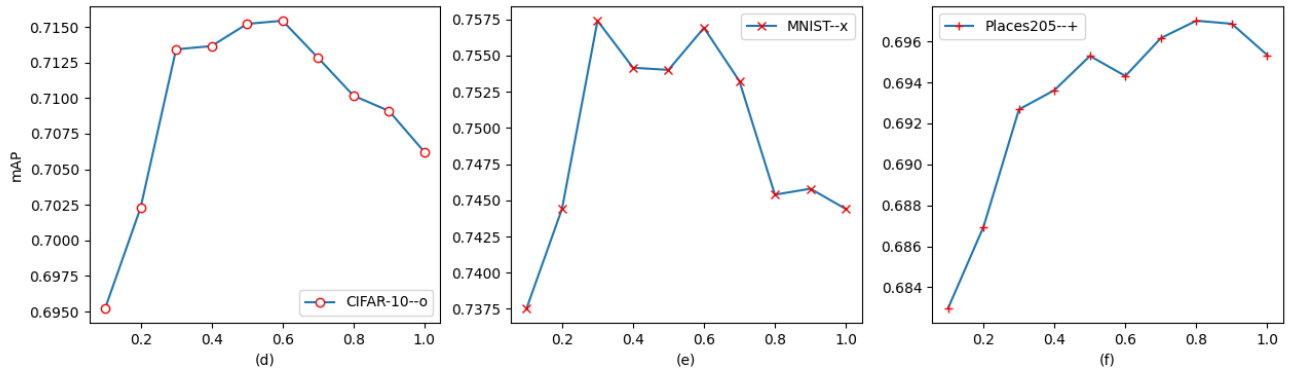

Fig. 9 Influence of parameter φ^t on the CIFAR-10, MNIST, and Places205 datasets.

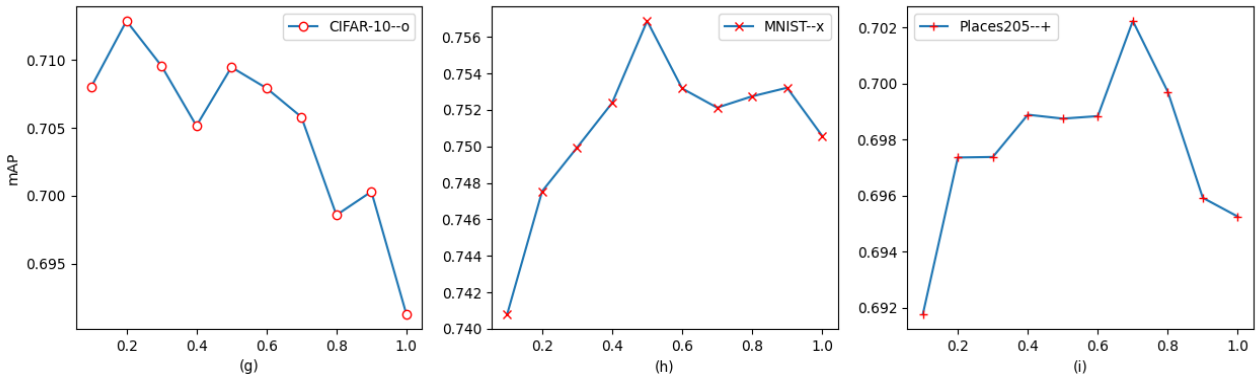

Fig. 10 Influence of parameter λ^t on the CIFAR-10, MNIST and Places205 datasets.

5. Conclusion

To solve the data imbalance problem, this paper proposes a novel supervised online hashing method, Label projection based on Hadamard codes for Online Hashing (LHOH), which jointly utilizes label projection and similarity preservation mechanism. The proposed LHOH uses Hadamard codes instead of hash codes as the target domain for label projection. As a result, we can quickly obtain a closed solution of the label projection matrix using the least squares regression method. Then, we solve the data imbalance problem of the similarity matrix between the old and new datasets by assigning label weight values through the label projection matrix. LHOH not only balances and maintains a similar relationship between new and old datasets but also makes full use of the label classes. To enhance the classification capability of online hash models, LHOH achieves triple supervision based on label classes by assigning Hadamard codes, label projection to the Hadamard codes matrix, and label embedding to similarity matrix learning, respectively. We conduct extensive experiments on three widely used datasets, CIFAR-10, MNIST, and Places205. The experimental results show that LHOH is superior to six current state-of-the-art online methods.

Acknowledgments

The authors express their gratitude to the institutions that supported this research: Shandong University of Technology (SDUT) and Jilin University (JLU).

References

[1] Fumin Shen et al., "Supervised Discrete Hashing," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 37-45, 2015. *Crossref,* https://doi.org/10.1109/cvpr.2015.7298598

[2] Jie Gui et al., "Fast Supervised Discrete Hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 40, no. 2, pp. 490-496, 2017. *Crossref,* https://doi.org/10.1109/tpami.2017.2678475

[3] Xin Luo, Ye Wu, and Xin-Shun Xu, "Scalable Supervised Discrete Hashing for Large-Scale Search," *Proceedings of the 2018 World Wide Web Conference*, pp. 1603-1612, 2018. *Crossref,* https://doi.org/10.1145/3178876.3186072

[4] Wei Liu et al., "Hashing With Graphs," *Proceedings of the 28th International Conference on Machine Learning*, vol. 12, no. 3, 2011.

[5] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang, "Semi-Supervised Hashing for Large-Scale Search," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 34, no. 12, pp. 2393-2406, 2012. *Crossref,* https://doi.org/10.1109/tpami.2012.48

[6] Jae-Pil Heo et al., "Spherical Hashing: Binary Code Embedding with Hyperspheres," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 37, no. 11, pp. 2304-2316, 2015. *Crossref,* https://doi.org/10.1109/tpami.2015.2408363

[7] Mengyang Yu, Li Liu, and Ling Shao, "Structure-Preserving Binary Representations for RGB-D Action Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 38, no. 8, pp. 1651-1664, 2015. *Crossref,* https://doi.org/10.1109/tpami.2015.2491925

[8] Wengang Zhou et al., "Scalable Feature Matching by Dual Cascaded Scalar Quantization for Image Retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 38, no. 1, pp. 159-171, 2016. *Crossref,* https://doi.org/10.1109/tpami.2015.2430329

[9] Hong Liu et al., "Towards Optimal Binary Code Learning Via Ordinal Embedding," in *Thirtieth AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, pp. 1258-1265, 2016. *Crossref,* https://doi.org/10.1609/aaai.v30i1.10167

[10] Xianlong Liu et al., "Multi-View Complementary Hash Tables for Nearest Neighbor Search," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1107-1115, 2015. *Crossref,* Https://Doi.Org/10.1109/ICCV.2015.132

[11] Li Liu, Mengyang Yu, and Ling Shao, "Multiview Alignment Hashing for Efficient Image Search," *IEEE Transactions on Image Processing,* vol. 24, no. 3, pp. 956-966, 2015. *Crossref,* Https://Doi.Org/10.1109/TIP.2015.2390975

[12] Fumin Shen et al., "Asymmetric Binary Coding for Image Search," *IEEE Transactions on Multimedia,* vol. 19, no. 9, pp. 2022-2032, 2017. *Crossref,* https://doi.org/10.1109/tmm.2017.2699863

[13] Long-Kai Huang, Qiang Yang, Wei-Shi Zheng, "Online Hashing," Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, pp. 1422-1428, 2013.

[14] Fatih Cakir, and Stan Sclaroff, "Adaptive Hashing for Fast Similarity Search," *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 1044-1052, 2015. *Crossref,* https://doi.org/10.1109/iccv.2015.125

[15] Fatih Cakir, Sarah Adel Bargal, and Stan Sclaroff, "Online Supervised Hashing," *Computer Vision and Image Understanding,* vol. 156, pp. 162-173, 2017. *Crossref,* https://doi.org/10.1016/j.cviu.2016.10.009

[16] Mingbao Lin et al., "Supervised Online Hashing via Hadamard Codebook Learning," *Proceedings of the 26th ACM International Conference on Multimedia*, pp. 1635-1643, 2018. *Crossref,* https://doi.org/10.1145/3240508.3240519

[17] Mingbao Lin et al., "Towards Optimal Discrete Online Hashing with Balanced Similarity," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 8722-8729, 2019. *Crossref,* https://doi.org/10.1609/aaai.v33i01.33018722

[18] Qing-Yuan Jiang, and Wu -Jun Li, "Asymmetric Deep Supervised Hashing," *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, pp. 3342-3349, 2018. *Crossref,* http://dx.doi.org/10.1609/aaai.v32i1.11814

[19] Y. Fang, H. Zhang, and L. Liu, "Label Projection Online Hashing for Balanced Similarity," *Journal of Visual Communication and Image Representation,* vol. 80, pp. 103314, 2021. Crossref, https://doi.org/10.1016/j.jvcir.2021.103314

[20] Yuzhi Fang, and Li Liu, "Scalable Supervised Online Hashing for Image Retrieval," *Journal of Computational Design and Engineering,* vol. 8, no. 5, pp. 1391-1406, 2021. *Crossref,* https://doi.org/10.1093/jcde/qwab052

[21] Cong Leng et al., "Online Sketching Hashing," *Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2503-2511, 2015. *Crossref,* https://doi.org/10.1109/cvpr.2015.7298865

[22] Xixian Chen, Irwin King, and Michael R. Lyu, "FROSH: Faster Online Sketching Hashing," in *UAI*, pp. 1-10, 2017.

[23] Yuzhi Fang, and Li Liu, "Angular Quantization Online Hashing for Image Retrieval," *IEEE Access,* vol. 10, pp. 72577-72589, 2022. *Crossref,* https://doi.org/10.1109/access.2021.3095367

[24] Edo Liberty, "Simple and Deterministic Matrix Sketching," *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 581-588, 2013. *Crossref,* https://doi.org/10.1145/2487575.2487623

[25] Yichao Lu et al., "Faster Ridge Regression via the Subsampled Randomized Hadamard Transform," *Advances in Neural Information Processing Systems,* vol. 26, pp. 369-377, 2013.

[26] Fatih Cakir et al., "Mihash: Online Hashing With Mutual Information," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 437-445, 2017. *Crosssref,* https://doi.ieeecomputersociety.org/10.1109/iccv.2017.55

[27] Zhenyu Weng, and Yuesheng Zhu, "Online Hashing With Bit Selection for Image Retrieval," *IEEE Transactions on Multimedia,* vol. 23, pp. 1868-1881, 2021. *Crossref,* https://doi.org/10.1109/tmm.2020.3004962

[28] R C. Veena, and S H. Brahmananda, "A Significant Detection of APT Using MD5 Hash Signature and Machine Learning Approach," *International Journal of Engineering Trends and Technology*, vol. 70, no. 4, pp. 95-106, 2022. *Crossref*, https://doi.org/10.14445/22315381/ijett-v70i4p208

[29] Mingbaea Lin et al., "Fast Class-Wise Updating for Online Hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 44, no. 5, pp. 2453-2467, 2022. *Crossref,* https://doi.org/10.1109/tpami.2020.3042193

[30] Leon Bottou, "Large-Scale Machine Learning with Stochastic Gradient Descent," in *Proceedings of COMPSTAT'2010: Springer*, pp. 177-186, 2010. *Crossref,* https://doi.org/10.1007/978-3-7908-2604-3_16

[31] Koby Crammer et al., "Online Passive Aggressive Algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551-585 2006.

[32] Robert E. Schapire, "Using Output Codes to Boost Multiclass Learning Problems," *Proceedings of the Fourteenth International Conference on Machine Learning*, vol. 97, pp. 313-321, 1997.

[33] J. Kittler et al., "Face Verification Using Error Correcting Output Codes," *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recogonition 2001*, vol. 1, pp. 7555-7560, 2001. *Crossref,* https://Doi.Org/10.1109/CVPR.2001.990552

[34] Jiayan Jiang, and Zhuowen Tu, "Efficient Scale Space Auto-Context for Image Segmentation and Labeling," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1810-1817, 2009. *Crossref,* https://doi.org/10.1109/cvpr.2009.5206761

[35] Hong Liu et al., "Ordinal Constraint Binary Coding for Approximate Nearest Neighbor Search," *IEEE Transactions on Pattern Analysis & Machine Intelligence,* vol. 41, no. 4, pp. 941-955, 2018. *Crossref,* https://doi.org/10.1109/tpami.2018.2819978

[36] Hong Liu et al., "Dense Auto-Encoder Hashing for Robust Cross-Modality Retrieval," *Proceedings of the 26th ACM International Conference on Multimedia,* pp. 1589-1597, 2018. *Crossref,* https://doi.org/10.1145/3240508.3240684

[37] A. Hedayat, and W. D. Wallis, "Hadamard Matrices and Their Applications," *The Annals of Statistics,* vol. 6, no. 6, pp. 1184-1238, 1978. *Crossref,* https://doi.org/10.1214/aos/1176344370

[38] Aristides Gionis, Piotr Indyk, and Rajeev Motwani, "Similarity Search in High Dimensions Via Hashing," *Proceedings of the 25th International Conference on Very Large Data Bases*, vol. 99, no. 6, pp. 518-529, 1999.

[39] Wang-Cheng Kang, Wu-Jun Li, and Zhi-Hua Zhou, "Column Sampling Based Discrete Supervised Hashing," *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, pp. 1230-1236, 2016. *Crossref,* https://doi.org/10.1609/aaai.v30i1.10176

[40] Karen Simonyan, and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *in International Conference on Learning Representations*, 2015.

[41] Y. Lecun et al., "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE,* vol. 86, no. 11, pp. 2278-2324, 1998. *Crossref,* https://doi.org/10.1109/5.726791

[42] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "Image net Classification With Deep Convolutional Neural Networks," *Advances in Neural Information Processing Systems,* vol. 60, no. 6, pp. 84-90, 2017. *Crossref,* https://doi.org/10.1145/3065386

[43] Mingbao Lin et al., "Hadamard Matrix Guided Online Hashing," *International Journal of Computer Vision,* vol. 128, no. 8, pp. 2279-2306, 2020. *Crosssref,* https://doi.org/10.1007/s11263-020-01332-z