Neurocomputing 636 (2025) 130031

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Broad hashing for image retrieval

Wing W.Y. Ng [a], Xuyu Liu [a], Xing Tian [b,*], Ting Wang [c,*], Jianjun Zhang [d,*], C.L. Philip Chen [a,e,f]

[a] Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[b] School of Artificial Intelligence, South China Normal University, Guangzhou, China
[c] College of Electronic Engineering, College of Artificial Intelligence, South China Agricultural University, Guangzhou, China
[d] College of Mathematics and Informatics, South China Agricultural University, Guangzhou, Guangzhou, China
[e] Navigation College, Dalian Maritime University, Dalian, China
[f] Brain and Affective Cognitive Research Center, Pazhou Lab, Guangzhou, China

ARTICLE INFO

Communicated by X. Zhu

Keywords:
Fast image retrieval
Broad learning
Supervised hashing
Incremental hashing

ABSTRACT

With the popularity of deep learning, deep hashing methods have become mainstream of hashing methods which adopt deep networks to learn better feature representation of images and simultaneously generate compact binary hash codes. Deep hashing methods have a bottleneck in training efficiency due to the complex structure of deep networks. In this work, we propose a broad hashing (BH) method with high retrieval performance and very short learning time. In BH, uncorrelated and balanced binary codes are assigned to each category through a Hadamard matrix. Then, a broad hashing network is constructed to learn hash functions which maps images to binary hash codes with high efficiency. Our method yields higher retrieval precision while its training time is 200 to 700 times faster than that of deep hashing methods. At the same time, more compact hash codes are obtained compared with conventional supervised learning methods. In addition, three incremental algorithms for BH are developed for dynamic environments, which enable the hash network to be remodeled without retraining. Experiments on three benchmark datasets validate the effectiveness and efficiency of BH.

1. Introduction

With the exponential growth of images and other multimedia data being uploaded to the Internet every second, content-based image retrieval (CBIR) has become a challenging task and attracted wide attentions[1]. Nearest neighbor search methods such as traversal-based and tree-based retrieval algorithms are not suitable for high-dimensional data space, so researchers focus on approximate nearest neighbor (ANN) search methods instead. Among ANN methods, hashing methods use less memory cost while yield higher retrieval efficiencies, thus they have been widely researched in CBIR task. They use a set of hash functions to map high-dimensional real-valued features of contents to compact binary hash sequences. The similarity between two images is measured by Hamming distances between binary sequences of images.

Early hashing research works on CBIR learn hash functions based on probability theories and ignore the distribution and semantic information of data. Locality sensitive hashing (LSH) [2] is one representative method that generates hash codes with random projections. To learn more informative hash functions, recent research works mainly focus on data-dependent hashing methods, which mainly consist of unsupervised and supervised hashing methods. Without leveraging label of images, unsupervised hashing methods [3–5] make use of data distribution information to learn hash functions. In contrast, supervised hashing methods adopt semantic information to improve retrieval performance, such as strongly constrained discrete hashing (SCDH) [6].

Aforementioned hashing methods without adopting deep learning network are named as conventional hashing methods. Conventional hashing methods works with handcrafted features (e.g., GIST [7]) while the limited semantic information of images reflected by handcrafted features becomes a bottleneck of image retrieval. The capability of deep learning network of extracting rich representative features from complex data structures relieves this bottleneck. Convolutional neural network hashing (CNNH) first introduces CNN [8] into the hashing algorithm [9], which is further developed by Lai et al. with using a triple similarity relationship [10]. Then, more in-depth research works on both unsupervised and supervised hashing are proposed. Supervised deep hashing methods learn image representations and

* Corresponding authors.
E-mail addresses: shawntian123@gmail.com (X. Tian), tingwang@ieee.org (T. Wang), jjzhangcs@gmail.com (J. Zhang).

https://doi.org/10.1016/j.neucom.2025.130031
Received 15 July 2023; Received in revised form 16 July 2024; Accepted 15 March 2025
Available online 24 March 2025

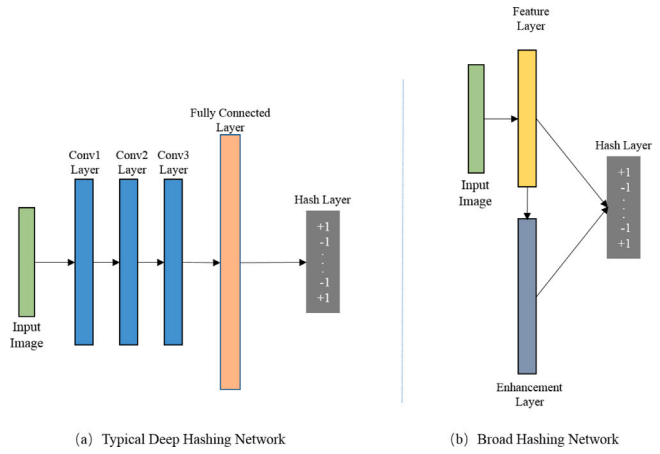

Fig. 1. The comparison of deep hashing and our proposed method. (a) The architecture of the typical deep hashing. (b) The architecture of the broad hashing.

hashing functions simultaneously, which output image representations and hashing codes at last two layers [11,12]. Without semantic information, some unsupervised deep hashing methods optimize their models via minimizing quantization loss [13] or data reconstruction loss [14], which fix their similarity graph matrixes. Other unsupervised deep hashing methods update similarity graph matrixes with learned deep models [15], or generate pseudo labels by clustering and train hash functions through adaptive similarity [16].

Many deep hashing methods have been proposed [17,18]. However, existing deep hashing methods generally suffer from low training efficiency due to the large number of connection weights in filters and layers [19]. Meanwhile, multiple epochs are always required for retraining to obtain high retrieval performance, which further reduces the training efficiency of hashing learning. Moreover, quantization loss is caused by the sigmoid-thresholding on the output of hash layer and will increase gradually due to the backpropagation in multiple epochs for deep hashing. Meanwhile, the backpropagation from the hash layer with short hash code length to a high-dimensional real-valued fully-connected layer leads to further loss of information and difficulty in deep hashing training [20].

To handle aforementioned drawbacks of deep hashing, a novel broad hashing (BH) method is proposed in this paper, which achieves comparable accuracy as deep hashing methods with extremely low complexity for training. To our best knowledge, BH is also the first attempt to introduce broad learning system (BLS) [19] to hashing-based image retrieval problems. Difference between architectures of deep hashing and broad hashing is presented in Fig. 1. In BH method, original inputs are projected to feature nodes and then expanded to enhancement nodes. The feature node and enhancement node denote the neuron in the feature layer and enhancement layer, respectively. A feature node is not a vector nor a scalar. Instead, it works effectively as a linear or nonlinear transformation function and it does not change the size of its output (whether the input is a vector or a matrix or a scalar). All of nodes are placed in one layer which avoids the computation of connection weights between multiple layers and the model retraining. With this structure, connection weights can be learned directly through a matrix computation, which leads to a high training efficiency. Moreover, BH utilizes a Hadamard matrix to generate ideal hashing codes and compose the hash layer directly. Hashing weights between a BH feature-extraction layer and an BH ideal hash layer can be computed using a matrix computation only without backpropagation, i.e., which do not cause accumulation of quantization loss

Contributions of this paper are summarized as follows:

1. A novel broad hashing (BH) method is proposed which yields high retrieval accuracy with significantly less training time comparing to existing deep hashing methods. As far as we know, this is the first work which introduces the broad learning system for hashing learning. The feature-extraction layer improves training efficiency greatly and does not need backpropagation in training. Moreover, the proposed method uses Hadamard matrix to generate compact hash codes. The matrix helps to train the hash functions uncorrelated and allows misclassifications of a few hash functions without affecting the final classification results.
2. Three incremental broad hashing methods are developed based on the original BH to train more effective hash functions under the complex and changing data environment. This shows the flexibility and scalability of our BH. These incremental BH hashing methods update the network rapidly by increasing BH network nodes, which stably improve the adaptability of BH for the current data environment without retraining on the previous data.
3. Experimental results on three public datasets show the effectiveness and efficiency of BH. The compact hash codes generated by BH yield higher retrieval accuracy than that of other state-of-the-art hashing methods, while its training time is 200 to 700 times faster than that of deep hashing methods.

The remainder of this paper is organized as follows. Section 2 gives a brief literature review about hashing methods and broad learning system. Section 3 describes details of the proposed BH. Section 4 evaluates the retrieval performance of BH on three public datasets. This work is concluded in Section 5.

2. Related works

Considering that our method is a supervised hashing method, we give a brief review of related works of supervised hashing in this section, including conventional supervised hashing and deep supervised hashing. Then, a brief introduction of broad learning system is given.

2.1. Supervised hashing

Compared with unsupervised hashing, supervised hashing methods are improved significantly because they benefit from auxiliary information. Auxiliary information generally has two kinds of expression used in the supervised hashing, one is using point-wise label information directly and another is using pairwise similarity information. Supervised methods using the label information directly are as follows. CCA-ITQ utilizes canonical correlation analysis (CCA) to obtain suitable projection directions for image descriptors and label respectively. These projection directions are supposed to maximize the correlation of the projected image vectors and projected label vectors [21]. Supervised discrete hashing (SDH) [22] evaluates differences between labels and linear classifications obtained from hash codes, then optimizes the loss function through cyclic coordinate descent. Based on SDH, robust supervised discrete hashing (RSDH) [23] introduces the Cauchy loss to reduce the quantization error and the affection of outliers. Supervised short-length hashing (SSLH) [24] adopts correntropy to enhance the robustness of the classifier and reconstructs the original features and hash codes mutually to reduce semantic loss in condition of the short-length of hash bits. Our proposed method introduces the Hadamard matrix which adopts labels as ground truths to obtain hash codes and avoid the correlation between hash bits.

On the other hand, pairwise similarity information is usually represented by a similarity matrix. Kernel-based supervised hashing (KSH) [25] makes an equivalence between Hamming distances and code inner products to train the hash codes effectively. Column sampling based discrete supervised hashing (COSDISH) [26] optimizes hash codes alternatively in a discrete way based on columns which sampled from

the similarity matrix in iteration. Supervised adaptive similarity matrix hashing (SASH) [27] learns the similarity matrix adaptively and optimizes the matrix with label correlations. Compared to the predefined similarity matrix, the adaptive similarity matrix indicates the relationship between the data in multi-label datasets and zero-shot datasets more accurately. Probability ordinal-preserving semantic hashing (POSH) [28] proposes a probabilistic similarity preservation framework to train hash functions based on the triplet ordinal similarity in a fast convergence speed. Relaxed Energy Preserving Hashing (REPH) [29] uses two-layer hash functions, energy preservation strategies and semantic reconstruction mechanisms to solve the problem of single-layer hash functions being difficult to fully preserve the inherent structure of data and severely losing low dimensional hash encoding information. Supervised Discrete Multiple-Length Hashing (SDMLH) [30] simultaneously learning multiple length hash codes to solves the problem of existing hash methods requiring separate training of different length hash encoding models. This is an innovative multi-length hash learning method. Hierarchical Hashing Learning(HHL) [31] solves the problem of existing hash methods ignoring the complex structural information and semantic hierarchy of the original features through strategies such as hierarchical hashing, regularization constraints, and bidirectional semantic representation, and improves the performance of ISC tasks.

2.2. Deep hashing

Recent years, deep learning-based hashing methods have been proposed to capture richer semantic features. CNNH first introduces CNN into the hash algorithm and uses two stages to train image representation and hash codes [9]. While two-stages pattern disallows feedbacks of image representation for learning better hash codes, later deep hashing methods put two stages in the same framework. They mostly use the last layer of the network as the hash layer to generate low-dimensional binary codes. Supervised learning based discrete hashing (SLDH) [32] introduces multilayer network to convert semantic feature of images into binary code and adopts ADMM in optimization to solve discrete constraints. Deep class-wise hashing (DCWH) [33] introduces a Gaussian distribution-based cubic constraint loss with label information instead of using pairwise loss to preserve semantic similarity. Deep central similarity hashing (DCSH) [34] considers both the hashing loss and the classification loss and utilizes CCA to formulate the final training loss. Deep balanced discrete hashing (DBDH) leverages a straight-through estimator for discrete gradient propagation rather than using the continuous relaxation process to avoid the quantization error [35]. Cosine metric supervised deep hashing with balanced similarity (BCMDH) [36] quantizes cosine distance entropy and makes an integration with contrastive cosine similarity, which helps the model to preserve the semantic information and reduce quantization loss.

Some deep hashing methods not only construct the loss function, but also build special network structures to make hash codes reflect more features. Deep discrete supervised hashing (DDSH) [37] learns feature representations and binary codes alternatively in iterations to enhance the feedback between deep features extraction procedures and discrete encoding procedures. Deep variational and structural hashing (DVStH) [20] designs variational blocks to obtain probabilistic latent representations and uses a struct layer instead of the hash layer to solve the bottleneck of accuracy under short-length hash bits. Siamese dilated inception hashing (SDIH) [38] leverages contextual information and category-level semantics to enhance intra-group correlation in its novel Siamese inception dilated network architecture. Unlike most deep hashing methods adopt AlexNet or ResNet as the backbone architecture, hashformer [39] and vision transformer hashing (VTS) [40] utilize a pre-trained vision transformer to extract image features. Fine-Grained Hashing (FISH) [41] through a dual filtering mechanism to solves the problems of fine-grained feature extraction and feature refinement and improve the training efficiency and performance of the model while maintaining semantic relationships between data instances.

2.3. Broad learning

However, while deep learning brings convenience to image retrieval, its shortcomings such as too many hyperparameters, complex network structures, occupying more memory, and slow running speed also limit the development of deep hashing. For drawbacks of deep learning in machine-learning tasks, Chen et al. propose a broad learning system (BLS) [19] on the basis of random vector functional-linked neural network (RVFLNN). The network of BLS has only one layer actually which consists of a feature layer and an enhancement layer so that training time is much lower than deep learning. Meanwhile, BLS can adjust the size of the model to adapt to the data by increasing feature nodes or enhancement nodes, without redesigning the network to achieve incremental learning.

The BLS attracts a lot of attention and derives several variants which obtain better image recognition performance [42]. Weighted BLS [43] leverages penalty factors for the noise and outliers of datasets to reduce their contribution to recognizing model. To solve the problems of rule explosion of common neuro-fuzzy approach, fuzzy BLS [44] is proposed which replaces feature nodes of the BLS with TS fuzzy subsystems and maps the intermediate output of fuzzy subsystems to the enhancement layer. In the procedure, K-means is adopted to calculate the number of fuzzy rules and the center of the Gaussian membership. On the basis of the homogeneous incremental algorithms in the original BLS, some research works put their attention on incremental learning. BLS with reinforcement learning signal feedback (BLRLF) [45] utilizes adaptive dynamic programming (ADP) to calculate near-optimal weights for increments and take outputs of network as feedbacks. The heuristic search method is used to optimize the network autonomously with integrating of the three basic incremental hashing. Moreover, unsupervised variants [46,47] and semi-supervised variants [48,49] have also been developed, which are more suitable for the real-word data distribution.

BLS has several advantages that enable it to be widely used. F. Hao et al. predict typhoon trajectories with a broad learning ensemble system, which is built with 10 BLS through bagging method [50]. Peng et al. introduce BLS to an edge computing-based traffic analysis system due to the peculiarity that incremental BLS supports inertial updates of model on edge nodes without training the whole dataset [51]. Hence, BLS has great advantages compared with conventional deep learning methods which also is potentially helpful for image retrieval based on hashing. There are also some limitations existing for BLS. First, BLS has simpler network structure and does not require learning weights, which may limit its ability to extract finer-grained information from training data. Second, with the number of enhancement nodes increases, training of BLS may become memory-intensive.

3. Broad hashing

In this section, broad hashing (BH) is introduced in detail. Section 3.1 introduces the general definition of hashing. Section 3.2 introduces the Hadamard matrix which generates the hash code table. Section 3.3 introduces the architecture of BH. Section 3.4 introduces the incremental algorithms of BH.

3.1. Problem definition

Given an image dataset $D = \{X, Y\}$, where $X = \{X_i\}_{i=1}^{n} \in \mathbb{R}^{n\times d}$ denotes a set of n d-dimensional training samples and $Y = \{y_i\}_{i=1}^{n} \in \mathbb{N}^{n}$ denotes corresponding labels of the training set. The proposed BH aims to find a set of hash functions to map high-dimensional images X onto a set of binary codes $B = \{B_i\}_{i=1}^{n} \in \{-1, +1\}^{n\times r}$, where $B_i = \{b_i^1, b_i^2, \ldots, b_i^r\}$ denotes the hash code of a sample. The discrete hash function is defined as follows:

$$B = sign(XW_h) \tag{1}$$

where $W_h \in \mathbb{R}^{d\times r}$ denotes the mapping matrix.

With hash codes, Hamming distance is calculated as follows to determine the similarity of two samples:

$$dist_{ij} = B_i \oplus B_j \tag{2}$$

where B_i and B_j are two rows of the hash table B while $\oplus$ stands for the exclusive OR operation. Hamming distances between similar samples are small while distances between dissimilar samples are large. As much as possible, we hope that the hash function obtained by BH can map similar samples to the same hash bucket in the Hamming space and vice versa. Samples in the same hash bucket share the same hash code.

3.2. Hadamard matrix for hash codes

Data of different categories is expected to be mapped to different hash buckets with different hash codes. The distance between two different hash buckets (i.e., the inter-class distance) equals to the Hamming distance between corresponding hash codes. When Hamming distances between dissimilar data are large enough, it is insignificant even if mapping errors happen to some hash bits. Therefore, we hope that the Hamming distances between dissimilar samples are as large as possible, i.e., hash buckets of different categories shall be separated as far as possible. Moreover, efficient hash codes should also have two characteristics, i.e., high variance of hash bits and hash bits are pairwise uncorrelated. When the variance of hash bits is high, the entropy of hash code is maximized which yields more informative hash codes. When hash bits are pairwise uncorrelated, the information redundancy of hash codes is minimized. To achieve aforementioned goals, Hadamard matrix [52] is employed to BH to generate r-bit effective hash codes for data according to their labels.

A Hadamard matrix H_r is an $r \times r$ matrix, with its elements being either -1 or $+1$. The product of H_r and its transpose satisfies $H_r H_r^T = H_r^T H_r = rI_r$, where I_r is an identity matrix of order r. The Off-diagonal elements of I_r equal to 0, which proves the orthogonality of both rows and columns of H_r. The orthogonality of both row vectors and column vectors of H_r makes sure that it fully satisfies the requirements of generating effective hash codes [53].

To generate suitable hash codes from Hadamard matrix, the order of matrix satisfies $r = \{r | r = 2^\tau, r \geq l\}$, where l denotes the number of categories of the dataset D and τ is a natural number. A discrete $r \times r$ Hadamard matrix can be constructed in the following manner:

$$H_r = H_{2^\tau} = \begin{bmatrix} H_{2^{\tau-1}} & H_{2^{\tau-1}} \\ H_{2^{\tau-1}} & -H_{2^{\tau-1}} \end{bmatrix} \tag{3}$$

with $H_1 = [1]$ and $H_2 = \begin{bmatrix} H_1 & H_1 \\ H_1 & -H_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$.

For a dataset D, a row of H_r can be used as a hash code for a category label. l rows of H_r are sampled to form the hash code table H_l. The Hamming distance between two hash code is calculated as follows:

$$dist_{ij} = \frac{1}{2}(r - \langle H_i, H_j \rangle) = \frac{r}{2} \tag{4}$$

$y_i \in [1, l]$, so the ideal hash code (U_i) of each sample (X_i) in category y_i are all assigned to be equal to the y_i row of H_l. Hamming distances between X_i and other samples in different categories are $r/2$ according to Eq. (4). As long as the number of bits where error occurs is less than a quarter of the length of hash code, the hash code of the sample is still the closest to the correct hash code and the sample can be still classified correctly.

3.3. Broad hashing architecture

A typical deep hash architecture consists of a basic neural network and a hash layer with the loss function. The basic network consists of a series of convolution layers, pooling layers and fully connected layers, which aims to learn abstract features from high-dimensional data. The hash layer is essentially a fully connected layer whose number of nodes is set as the number of hash bits. Sigmoid function is used to binarize the output of the hash layer for getting the final binary codes. The whole network is optimized through backpropagation with the loss function to yield a high accuracy. However, the backpropagation mechanism and the huge number of hyperparameters prolong the training time and occupy more space to save the model. Meanwhile, the hash layer that follows the length of the required short binary code makes the hash layer become a bottleneck of the retrieval precision and the backpropagation is handicapped due the dependence of loss function on the hash layer. Our method uses a special broad structure BLS to extract features, which ensures both fast training speed and high retrieval precision even with a short hash code length.

Different from the conventional deep network that requires a multi-layer network, BLS combines a feature layer and an enhancement layer into a new input layer. Thus, BH network weights, i.e., the BH hash function W_h, are obtained by a one-step training with hash codes we obtained through the Hadamard matrix as the output layer of the network. For a given dataset $D = \{X, Y\}$, the description of a discrete BH architecture is shown as follows.

X is randomly mapped a times and each mapping generates α nodes to yield the shallow feature of the training set first. Let f and g be activation functions. For each mapping, X is mapped onto S_i using W_{si} and β_{si} and all the shallow-mapped nodes are denoted as shallow-mapped layer $S^a = [S_1, \ldots, S_a]$, which is represented as follows:

$$S_i = f(XW_{si} + \beta_{si}), i = 1, \ldots, a \tag{5}$$

To get richer information of training data, shallow-mapped nodes are enhanced c times with randomly generated weights W_{ej} and bias β_{ej}. Enhancement nodes are calculated as follows:

$$E_j = g(S^a W_{ej} + \beta_{ej}), j = 1, \ldots, c \tag{6}$$

All enhancement nodes are denoted as enhancement layer $E^c = [E_1, \ldots, E_c]$. We denote the concatenation of the shallow-mapped layer S^a and the enhancement layer E^c as full feature layer ($F_a^c = [S^a | E^c]$). The binary hash code output (B) is computed as follows:

$$\begin{aligned} B &= sign([S_1, \ldots, S_a | E_1, \ldots, E_c] W_h) \\ &= sign([S^a | E^c] W_h) \\ &= sign(F_a^c W_h) \end{aligned} \tag{7}$$

where sign(·) is a binarization function. To make the mapping matrix W_h more informative, B is expected to be as similar as possible with U which is obtained through H_l. Thus, we set U as the values in hash layer BH. The detailed architecture of BH is shown in Fig. 2. The objective of hash layer is to learn the relationship between the training sample X_i and its corresponding binary code U_i according to the label y_i. We use the following loss function to calculate the network weights W_h to reduce the quantization error caused by the conversion of data from high-dimensional space to binary Hamming space. In other words, the loss caused by sign(·) is minimized and the original locality structure of the data is retained as much as possible.

$$\arg\min_{W_h} \|U - F_a^c W_h\|_2^2 + \lambda \|W_h\|_2 \tag{8}$$

where λ denotes a regularization parameter. Through the ridge regression theory, W_h is computed as follows:

$$W_h = (\lambda I + F_a^c F_a^{cT})^{-1} F_a^{cT} U \tag{9}$$

Detailed procedures of BH are shown in Algorithm 1.

Feature learning of BH depends on numbers of shallow-mapped layer nodes and enhancement layer nodes. Generally speaking, more nodes in the full feature layer and the wider the trained network, the more detailed the network for feature mining, the higher the accuracy for image retrieving, and the longer the training time. Fortunately, the training time of BH is generally much shorter than that of deep hashing algorithms since no feedback of learning is needed in the BH architecture.

Algorithm 1 BH: Broad Hashing

Input Training Dataset $D = \{X, Y\}$ with n samples and l categories, the length of hash codes r

Output The BH network parameter W_h

Construct a $r \times r$ Hadamard matrix H_r
Sample l rows of H_r as a hash code table H_l
Initialize the node parameters of mapping a, c
for $i = 0; i \le a$ **do**
    Randomly generate W_{si} and β_{si}
    Compute S_i via Eq. (5)
end for
Denote the shallow-mapped layer as $S^a = [S_1, \cdots, S_a]$
for $j = 0; j \le c$ **do**
    Randomly generate W_{ej} and β_{ej}
    Compute E_j via Eq. (6)
end for
Denote the enhancement layer as $E^c = [E_1, \cdots, E_c]$
Denote the full feature layer as $F_a^c = [S^a | E^c]$
According the label y_i assign hash code U_i from H_l to each sample X_i, denote $U = [U_1, \cdots, U_n]$
Calculate the network weights W_h via Eq. (9)
return W_h

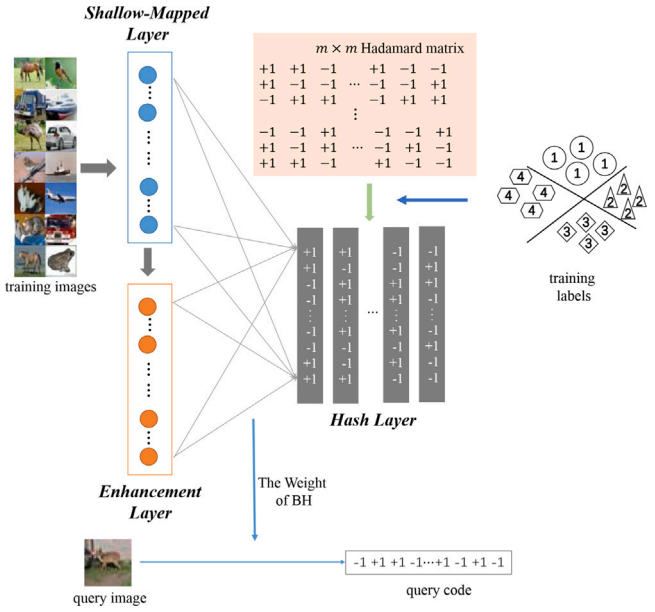

Fig. 2. The BH architecture.

3.4. Incremental broad hashing

In this section, the proposed BH is extended for incremental hashing. According to the architecture of BH, three incremental hashing methods are proposed: incremental hashing of shallow-mapped nodes, enhancement nodes, and new incoming data, respectively. The first two methods adjust the current BH network through incrementing either shallow-mapped nodes or enhancement nodes dynamically. The incremental hashing of new incoming data is proposed to adapt new incoming data for the trained network that simulate real data environments. All three methods train the updated part in each incremental step only to avoid repeated training.

Architectures of the three incremental BH methods are shown in Fig. 3 and descriptions of these three methods are presented as follows.

(1) Incremental Hashing of Enhancement Nodes

The newly added c_0 groups of enhancement nodes are expressed as follows:

$$E^{c_0} = [g(S^a W_{e_{new}k} + \beta_{e_{new}k})], k = 1, \ldots, c_0 \tag{10}$$

where $W_{e_{new}k}$ and $\beta_{(e_{new}k)}$ denote newly generated weights and bias, respectively. The newly formed enhancement layer is denoted as follows:

$$E^{c_2} = [E^c | E^{c_0}] \tag{11}$$

where c_2 denotes the number of groups of enhancement nodes on the newly formed enhancement layer and $c_2 = c + c_0$. The new full feature layer is denoted as follows:

$$F_a^{c_2} = [S^a | E^{c_2}] = [F_a^c | g(S^a W_{e_{new}k} + \beta_{e_{new}k})] \tag{12}$$

Thus, the pseudoinverse matrix of $F_a^{c_2}$ is computed as follows:

$$(F_a^{c_2})^+ = [((F_a^c)^+ - p\omega^T)^T, \omega]^T \tag{13}$$

where $p = (F_a^c)^+ g(S^a W_{e_{new}k} + \beta_{e_{new}k})$,

$$\omega^T = \begin{cases} z^+ & if\ z \neq 0 \\ (1 + p^T p)^{-1} p^T (F_a^c)^+ & if\ z = 0 \end{cases}$$

and $z = g(S^a W_{e_{new}k} + \beta_{e_{new}k}) - F_a^c p$.

Updated weights of the BH network under the expansion of enhancement layer are computed as follows:

$$W_a^{c_2} = [W_a^c - p\omega^T U, \omega^T U]^T \tag{14}$$

where $W_a^{c_2}$ and W_a^c denote the new W_h after adding enhancement nodes and the mapping matrix W_h computed with F_a^c and U, respectively.

(2) Incremental Hashing of Shallow-Mapped Nodes

The incremental of shallow-mapped nodes helps to obtain sufficient shallow feature information when the retrieval performance is not improved after increasing enhancement nodes. Newly added a_0 groups of shallow-mapped nodes are denoted as follows:

$$S^{a_0} = [f(X W_{s_{new}t} + \beta_{s_{new}t})], t = 1, \ldots, a_0 \tag{15}$$

where $W_{s_{new}t}$ and $\beta_{s_{new}t}$ denote newly generated weights and bias, respectively. The newly formed shallow-mapped layer S^{a_1} is denoted as $S^{a_1} = [S^a | S^{a_0}]$, where a_1 denotes the number of groups of nodes on the newly formed shallow-mapped layer.

Correspondingly, new shallow-mapped nodes S^{a_0} are enhanced c times according to the Eq. (6). Thus the new enhancement layer is denoted as $E^{c_{new}} = [E^c | E'^c]$, where E'^c denotes newly enhancement nodes.

Then, the full feature layer is rewritten as $F_{a_1}^{c_{new}} = [S^{a_1} | E^{c_{new}}] = [F_a^c | S^{a_0} | E'^c]$. The pseudoinverse matrix of $F_{a_1}^{c_{new}}$ is computed as follows:

$$(F_{(a_1)}^{c_{new}})^+ = [((F_a^c)^+ - p_1 \omega_1^T)^T, \omega_1]^T \tag{16}$$

where $p_1 = (F_a^c)^+ [S^{a_0} | E'^c]$,

$$\omega_1^T = \begin{cases} z_1^+ & if\ z_1 \neq 0 \\ (1 + p_1^T p_1)^{-1} p_1^T (F_a^c)^+ & if\ z_1 = 0 \end{cases}$$

and $z_1 = [S^{a_0} | E'^c] - F_a^c p_1$.

Thus, updated weights of the BH network under the expansion of the shallow-mapped layer are computed as follows:

$$W_{a_1}^{c_{new}} = [W_a^c - p_1 \omega_1^T U, \omega_1^T U]^T \tag{17}$$

where $W_{a_1}^{c_{new}}$ denotes the new W_h after adding shallow-mapped nodes and enhancement nodes.

(3) Incremental Hashing of New Data

The two incremental methods mentioned above improve the adaptability and learning capability of the network for the data distribution by adding an appropriate number of nodes to modify BH. The third incremental method of BH is designed for dynamic data environments. For deep hashing methods, the entire model needs to be retrained for

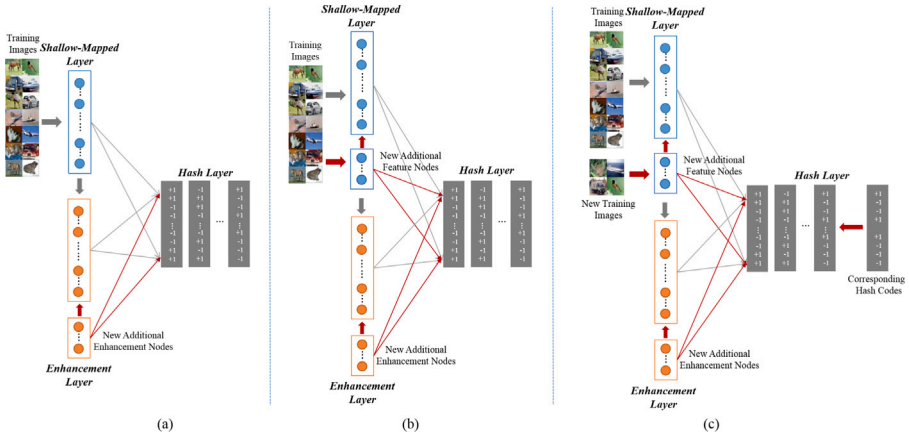

Fig. 3. The architectures of three incremental BH. (a) The incremental BH of enhancement nodes. (b) The incremental BH of feature nodes. (c) The incremental BH of data.

the arrival of any new data. The incremental BH of new data learns features of the new data on the basis of weights of learned BH without retraining, which provides better adaptability to the dynamic data.

New data is denoted as X_θ and randomly mapped to the shallow-mapped layer and the enhancement layer. The increment of full feature layer is formulated as follows:

$$F_{a_\theta}^{c_\theta} = [S^{a_\theta} | E^{c_\theta}] \tag{18}$$

where S^{a_θ} and E^{c_θ} denote new shallow-mapped nodes and new enhancement nodes, respectively. The associated full feature layer is set as $F_{a_3}^{c_3} = [F_a^c | F_{a_\theta}^{c_\theta}]$, where $c_3 = c + c_\theta$ and $a_3 = a + a_\theta$. The pseudoinverse matrix of $F_{a_3}^{c_3}$ is computed as follows:

$$(F_{a_3}^{c_3})^+ = [((F_a^c)^+ - p_2\omega_2^T)^T, \omega_2]^T \tag{19}$$

where $p_2 = (F_a^c)^+ F_{(a_\theta)}^{(c_\theta)}$ and

$$\omega_2^T = \begin{cases} z_2^+ & if\ z_2 \neq 0 \\ (1 + p_2^T p_2)^{-1} p_2^T (F_a^c)^+ & if\ z_2 = 0 \end{cases}$$

and $z_2 = F_{(a_\theta)}^{c_\theta} - F_a^c p_2$. Thus, updated weights of the BH network are computed as follows:

$$W_{a_3}^{c_3} = [W_a^c - p_2\omega_2^T U, \omega_2^T U]^T \tag{20}$$

where $W_{a_3}^{c_3}$ denotes the new W_h after adding new data.

4. Experiment

In this section, experiments on three benchmark datasets are performed to evaluate the performance of BH against to several existing hashing methods.

4.1. Datasets

Three commonly used datasets, i.e. MNIST, CIFAR10 and Caltech256, are utilized as benchmark for performance evaluation. These three datasets are widely used for performance evaluation by existing broad learning methods [45] and hashing retrieval methods [29,30]. Meanwhile, these datasets are also single-labeled datasets, which make it easier for experimental results analysis.

CIFAR10 dataset consists of 60,000 images with a size of 32 × 32 extracted from 80M Tiny Images. The dataset is manually labeled and divided into 10 categories. In our experiments, BH is compared with conventional hashing algorithms and deep hashing algorithms respectively. According to the dataset setting of these comparative methods in their original papers, we employ two different representations: 512-dimensional GIST feature vectors for conventional hashing algorithms and 4096-dimensional CNN features vectors for deep hashing algorithms. For both conventional hashing algorithms and deep hashing algorithms, we split 59,000 instances as the training set and 1000 instances as the query set.

MNIST dataset consists of 70,000 handwritten digital images represented by 784-dimensional original pixels. Images in the dataset are classified into 10 categories and labeled from 0 to 9. In our experiments, 60,000 instances are randomly selected as the training set and the remaining 10,000 instances are used as the query set.

Caltech256 dataset consists of 30,607 images which are extracted from Google Image dataset. The dataset is classified into 256 categories and there are at least 80 images in each category. We use 1024-dimensional CNN feature vectors to represent images and randomly split 28,047 instances as the training set and 2560 instances as the query set.

4.2. State-of-the-art methods

To evaluate the performance of BH, we select representative hashing algorithms as comparative methods, i.e., LSH [2], ITQ [21], SDH [22], FSDH [54], SCDH [6], COSDISH [26], RLPSDH [55], HRNH [56], DDSH [37], DCH [57], DPN [58], DBDH [35], DVStH [20], SDIH [38], SLDH [32], and BCMDH [36]. The first seven are conventional supervised hashing methods without deep network while the other nine are deep hashing methods. To ensure fair comparisons, most of hyper-parameters of comparative methods keep the original setting. For conventional supervised methods, we reproduce them with source codes provided in the papers and keep all of default or optimal settings. Specifically, for SDH, FSDH, and SCDH, we employ 1000 instances as anchors. For deep hashing methods, we re-run experiments of DCH, DBDH, DPN, and DDSH with corresponding default settings provided in their papers or source codes. Retrieval results of the rest of methods are referring to the original papers directly.

4.3. Implementation setting

To construct a BH framework, three parameters δ_1, δ_2, and δ_3 are used to determine the composition of the full feature layer, whose effect are equivalent to node parameters of mapping a, c in Algorithm 1. δ_1, δ_2, and δ_3 denote the number of shallow-mapped nodes per group, the number of groups of shallow-mapped nodes, and the number of enhancement nodes, respectively. The setting of δ_1, δ_2, δ_3 for different datasets are shown in Table 1. W_s and W_e are randomly generated which denote mapped weights of shallow-mapped layer and enhancement layer, respectively. Moreover, two parameters γ and s are used to optimal BH framework. Parameter s is the shrinkage parameter which is set to be 0.8 on CIFAR10 with CNN feature and 0.2 on the rest datasets.

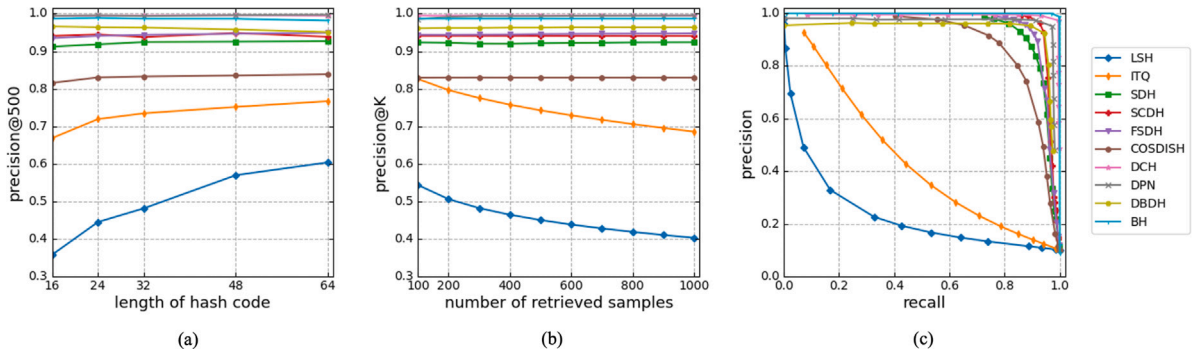

Fig. 4. Precision results of MNIST. (a) Precision@500 curves with different lengths of hash codes on MNIST. (b) Precision@K curves using 32-bit binary hash codes on MNIST. (c) Recall-Precision curves using 32-bit binary hash codes on MNIST.

Table 1
The setting of network parameters of BH.

	MNIST	CIFAR10	CIFAR10-GIST	Caltech256
δ_1	100	100	80	200
δ_2	50	50	30	30
δ_3	20,000	5000	50,000	20,000

The regularization parameter γ is set to be 2^{-28} on CIFAR10 with CNN feature and 2^{-27} on the rest datasets. All experiments are repeated 10 times to get the average performance as the final experimental results.

4.4. Retrieval metrics

MAP and Precision@K are used to evaluate the retrieval performance. MAP is the mean of average precisions of queries in the query set which is computed as follows:

$$MAP = \frac{1}{Q}\left(\sum_{q}^{Q}\frac{1}{R_q}\left(\sum_{i}^{N}\frac{R_{qi}}{i}\times rel_{qi}\right)\right) \tag{21}$$

where Q, N, R_q, and R_{qi} denote the size of query sets, the size of the dataset, the number of relevant images in the dataset of qth query, and the number of relevant images in top i retrieval results of qth query, respectively. $rel_{qi} = 1$ if the ith retrieved image is similar to qth query and 0 otherwise.

Precision@K is the average accuracy of the top K retrieved images over all queries and computed as follows:

$$Precision@K = \frac{1}{Q}\left(\sum_{q}^{Q}\frac{\sum_{j=1}^{K} rel_{qj}}{K}\right) \tag{22}$$

where K denotes the number of retrieved images, $rel_{qj} = 1$ if the jth image in top K retrieval results is similar to qth query and 0 otherwise.

4.5. Retrieval results on MNIST

Table 2 shows MAP values of different methods using different hash code lengths on the MNIST dataset. The last column of Table 2 shows the average rank of MAP for each hashing methods using different hash code lengths. The last row shows MAP values with standard deviation of the BH. Bolded texts in tables indicate the best performance of the corresponding metric for a dataset. Fig. 4(a), (b), and (c) report Precision@500 curves within different hash code lengths, Precision@K curves using 32-bit binary hash codes, and recall-precision curves using 32-bit binary hash codes, respectively.

From Table 2, MAP of BH yields 98.60%, 98.54%, 98.71%, 98.52%, and 98.62% with 16, 24, 32, 48, and 64-bit hash codes, respectively. Compared with conventional hashing methods, BH demonstrates its superiority on both MAP and top-K precision. BH also yields better performance comparing with some deep learning-based methods

Table 2
MAP with different hash code lengths on MNIST dataset.

Method	16-bit	24-bit	32-bit	48-bit	64-bit	Rank
Deep hashing						
SDIH	98.27	98.33	98.54	98.68	98.84	3.6
DBDH	96.30	95.9	96.40	96.00	95.50	6.2
DCH	**99.03**	99.12	**99.24**	**99.35**	**99.48**	**1.2**
DPN	98.43	**99.21**	99.01	99.15	99.14	2
HRNH	98.15	98.23	98.35	98.47	98.66	4.8
SLDH	87.72	–	94.76	–	96.87	8.3
Conventional hashing						
LSH	20.00	23.87	25.21	29.98	31.93	13.4
ITQ	40.40	43.14	43.17	44.18	45.12	12.4
SDH	89.35	90.48	91.28	91.94	92.42	10.2
COSDISH	84.36	85.91	86.34	86.82	87.05	11.4
FSDH	92.06	93.05	93.69	94.13	94.71	9.2
RLPSDH	92.83	–	94.09	94.99	95.04	8.25
SCDH	94.98	95.39	94.78	95.88	94.79	7.4
Our method						
BH	98.60 ±0.10	98.54 ±0.12	98.71 ±0.07	98.52 ±0.16	98.62 ±0.10	3.4

Table 3
Comparison between BH with deep hashing using 32 bits on MNIST dataset.

Method	MAP	Precision@500	Time (s)
DCH	**99.24**	**99.5**	5133.30
DPN	99.01	98.76	8427.42
DBDH	96.4	95.8	11,940.00
BH	98.71	98.66	**800.52**

(e.g., SDIH, HRNH, SLDH), which may because our BH extracts deeper semantic features through a sufficiently wide enhancement layer. But for DCN and DPN, their performances are comparable to or even better than BH. There are two reasons: First, images of MNIST dataset are all grayscale images and relatively simple. These deep hashing extract features on the basis of VGG16. After multiple epochs, the semantic information of the dataset has been extracted to a very deep level. Therefore, deep hashing generally performs well on MNIST datasets. Second, due to the server memory limitation, the process of obtaining pseudo-inverse in the BH involves a lot of matrix operations, so the number of nodes in our shallow-mapped layer is relatively limited, which confines the image retrieval precision. Although the MAP of these deep hashing methods on MNIST reach higher than 99%, Table 3 shows that they use 6.41 to 14.92 times longer training time than that of BHs due to multi-layer network and too many parameters of deep hashing. In conclusion, BH yields comparable retrieval results using a shorter training time and is more suitable in real life problems.

Table 4
MAP with different lengths of hash codes on CIFAR10 dataset.

Method	16-bit	24-bit	32-bit	48-bit	64-bit	Rank
Deep hashing						
SDIH	72.2	73.58	74.75	74.83	75.18	5.2
DVStH	69.55	70.04	70.62	71.85	72.49	6.6
DBDH	65.7	68.6	67.92	69.7	69.82	8.4
DCH	70.21	70.89	70.07	70.53	69.97	6.8
DPN	76.18	77.74	78.49	78.9	78.28	4.2
DDSH	80.33	82.6	82.78	82.17	81.47	2.6
HRNH	65.76	67.84	67.85	68.29	68.85	9
SLDH	65.57	–	81.39	–	**87.62**	4.7
BCMDH	78.05	–	80.42	82.75	83.06	3.3
BH with CNN	**83.76**	**83.65**	**84.14**	**83.07**	84.15	**1.2**
	±0.85	**±0.84**	**±1.01**	**±0.75**	±0.90	
Conventional hashing						
LSH	11.73	13.55	13.76	14.23	14.58	7
ITQ	16.2	16.81	16.92	17.22	17.54	6
SDH	36.38	41.29	42.46	44.38	44.91	4.6
COSDISH	54.07	57.29	58.27	61.37	64.13	2.8
FSDH	37.52	40.92	42.56	43.93	45.13	4.4
SCDH	64.68	61.63	67.24	61.79	57.07	2.2
BH with GIST	**67.22**	**66.29**	**67.49**	**65.91**	**67.22**	**1**
	±0.93	**±1.26**	**±1.34**	**±1.16**	**±1.88**	

4.6. Retrieval results on CIFAR10

Table 4 shows MAP of different methods using different numbers of hash bits on the dataset CIFAR10. The last column of Table 4 shows the average rank of MAP using different numbers of bits compared with deep hashing and conventional hashing, respectively. Figs. 5 and 6 show precision curves of CIFAR10 dataset with 512-dimensional GIST features and with 4096-dimensional CNN features. Fig. 5(a), (b), and (c) show Precision@500 curves using different hash code lengths, Precision@K curves using 32-bit hash codes, and recall-precision curves using 32-bit hash codes. For fairness, we use the 4096-dimensional CNN features dataset to train BH (BH with CNN) compared with deep hashing methods and use the 512-dimensional GIST features dataset to train BH (BH with GIST) compared with conventional hashing methods. The training and the query sets consist of 59,000 and 1000 images, respectively.

According to the Table 4, based on the GIST feature, the MAP of BH is better than all other conventional hashing methods. The average rank of BH gets the first place. Meanwhile, as shown in the Fig. 5, BH yields the best precision performance compared with other comparative hashing methods. The board structure leads BH extract deeper semantic features from the dataset while these conventional hashing methods train hash functions with shallower GIST features, which leads BH outperform than other conventional hashing methods. Based on the 4096-dimensional CNN feature dataset, the performance of BH is significantly better than that of deep hashing on MAP, especially when the number of bits is small. According to the Fig. 6, BH also yields the best precision performance. The reason is that our BH obtains semantic features directly without complex multiple layers like deep hashing. It avoids multiple transmissions of redundant information between multiple connecting layers in epochs and losses of features led by some implementations such as dropout.

In order to make a concrete comparison with deep hashing methods, MAP, Precision@500, and training time of deep hashing methods and our method are presented in Table 5. It can be concluded that on 4096-dimensional features of CIFAR10, BH adapts quickly to the data distribution using only 8 s and yields a high performance, which is 267.08 to 420.00 times faster than deep hashing methods.

Fig. 7 provides top 20 retrieval results of comparative methods and BH, respectively, which show that BH outperforms other conventional and deep hashing methods. In Fig. 7(a) and (b), most conventional hashing methods retrieve a lot of irrelevant images (marked by red boxes) while BH retrieves correct results, for both "track" and "bird" queries. In Fig. 7(d), all five methods perform well for the query of "track" image. However, in queries of "ship" and "bird" images being shown in Fig. 7(c) and (e), four deep hashing methods retrieve irrelevant images while all retrieved images of BH are relevant.

Table 5
Comparison between BH with deep hashing using 32 bits on CIFAR10 dataset.

Method	MAP	Precision@500	Time (s)
DCH	70.07	73.47	2414.39
DDSH	82.78	80.51	2275.50
DPN	78.49	82.81	3578.34
DBDH	67.92	67.42	2773.00
BH with CNN	**84.14**	**86.8**	**8.52**

Table 6
MAP with different hash code lengths on Caltech256 dataset compared with conventional methods.

Method	64-bit	128-bit	256-bit	Rank
LSH	16.79	24.71	30.96	7
ITQ	30.96	37.13	39.59	6
SDH	46.47	50.49	53.83	4.3
COSDISH	56.52	67.78	72.79	2.3
FSDH	46.49	50.41	53.25	4.6
SCDH	**62.17**	67.15	69.77	2.3
BH	57.43 ± 1.34	**75.82 ± 0.75**	**77.98 ± 0.68**	**1.3**

Table 7
MAP and training time of hash codes on Caltech256 dataset.

Method	64-bit	Time (s)	128-bit	Time (s)	256-bit	Time (s)	Rank
COSDISH	56.52	71.46	67.78	340.2	72.79	2744.70	3
SCDH	**62.17**	22.82	67.15	22.68	69.77	59.96	**2.2**
DCH	60.31	11,957.32	59.53	13,820.48	60.32	13,546.74	4.5
DPN	57.49	3525.43	64.34	3793.70	63.8	4752.00	3.8
BH	57.43	**20.11**	**75.82**	**20.11**	**77.98**	**40.01**	**1.5**

4.7. Retrieval results on Caltech256

Different from the previous two datasets, Caltech256 has more categories but less images in each category. For such a dataset, it is better to use longer hash code to map different categories into different hash buckets. Table 6 shows the MAP of BH and conventional hashing methods using 64, 128, and 256-bit hash codes. Table 7 presents both MAP and training time compared with two deep methods and two outstanding conventional methods. Both last columns of two tables show the average rank of MAP over different number of bits of all hashing methods. Bolded texts in tables indicate the best retrieval performance or shortest training time.

Table 6 shows that our BH performs better than all conventional hashing methods. Though MAP of SCDH yields 62.17% with 64-bit hash code, which is about 5% higher than BH, the average rank of BH using different number of hash bits in the Table 6 gets the first place. To further compare the training efficiency with conventional methods, the training time of CODSISH and SCDH is given in Table 7. CODSISH yields nice retrieval accuracies but its training time also increases as the number of bits increases. SCDH costs a short training time while BH shows higher training efficiency than SCDH in most of the cases. Deep hashing methods are time-consuming for long hash codes which are exhibited in Table 7. For two deep hashing methods which reduce the classification loss through the error propagation of the deep network, both of them perform well in terms of retrieval accuracy but require very long training time. In the worst case, the training time of DCH is 687.24 times longer than that of BH's.

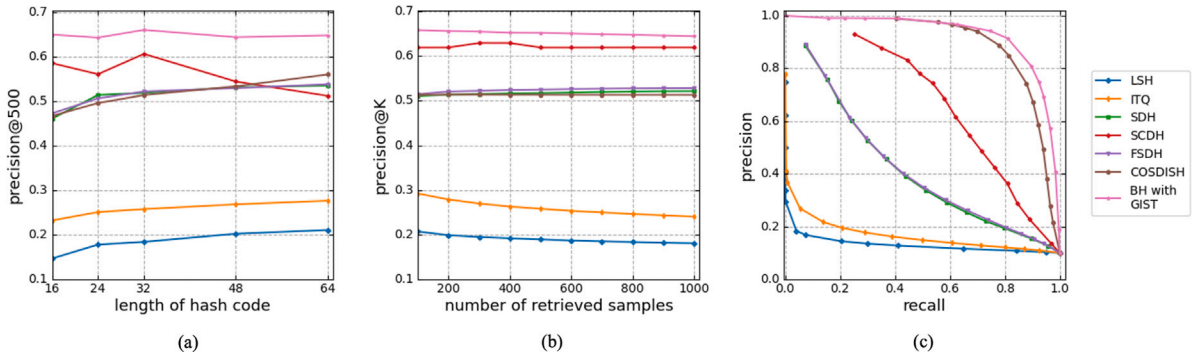

Fig. 5. Precision results of Cifar10 of 512-dimensional GIST features compared with conventional methods. (a) Precision@500 curves with different lengths of hash codes. (b) Precision@K curves using 32-bit binary hash codes. (c) Recall-Precision curves using 32-bit binary hash codes.

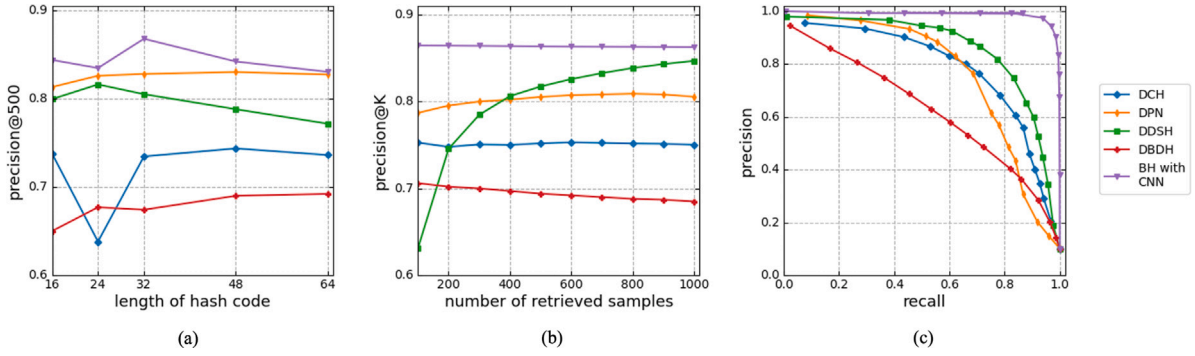

Fig. 6. Precision results of Cifar10 of 4096-dimensional CNN features compared with deep methods. (a) Precision@500 curves with different lengths of hash codes. (b) Precision@K curves using 32-bit binary hash codes. (c) Recall-Precision curves using 32-bit binary hash codes.

Fig. 7. Comparison of top 20 retrieved images between the state-of-the-art methods and our method. (a) Top 20 retrieved images for "SHIP" query compared with conventional hashing. (b) Top 20 retrieved images for "TRACK" query compared with conventional hashing. (c) Top 20 retrieved images for "SHIP" query compared with deep hashing. (d) Top 20 retrieved images for "TRACK" query compared with deep hashing. (e) Top 20 retrieved images for "BIRD" query compared with deep hashing.

Compared with them, BH yields excellent performance using much shorter training time. For 128-bit and 256-bit hash codes, retrieval performances of BH are better than all other methods. In the case of 64-bit hash code, BH is slightly inferior to some methods. It is because this dataset consists of 256 classes which requires a 256-dimensional Hadamard matrix to assign hash codes to each class. Hence, in the case of a short hash code, such as 64-bit, it is necessary to compress the binary code obtained from the Hadamard matrix which makes the final BH code table H_l unbalanced. This harms the retrieval performance of BH.

4.8. Experiments of incremental broad hashing

In this section, the 512-dimensional Cifar10 dataset is used to conduct experiments on incremental BH of enhancement nodes, full feature layer, and new data, respectively, to validate the adaptability of BH with respect to dynamic environments.

For the first two experiments, the hash code length is set as 32, parameters δ_1, δ_2, and δ_3 are set as 100, 20, and 10,000, respectively. Moreover, the number of additional shallow-mapped nodes per incremental step m_1, the number of additional enhancement nodes of the incremental shallow-mapped nodes per increment step m_2, and the

Table 8
The experiment on the incremental of enhancement nodes using 32-bit.

Number of incremental nodes	MAP@50	Precision@500	Time (s)
Initial	62.13	60.25	109.57
m_3	61.85	60.42	27.27
$2m_3$	62	60.29	30.55
$3m_3$	62.71	60.76	34.1
$4m_3$	62.97	60.92	37.49
$5m_3$	62.73	61.36	38.9
$6m_3$	63.38	61.73	44.09

Table 9
The experiment on the incremental of full feature nodes using 32-bit.

Number of new feature nodes	MAP@50	Precision@500	Time (s)
Initial	62.29	60.33	92.66
$m_1 \times m_2 + m_3 + m_1$	62.75	60.3	37.55
$2(m_1 \times m_2 + m_3 + m_1)$	63.43	61.48	36.35
$3(m_1 \times m_2 + m_3 + m_1)$	63.58	61.58	40.45
$4(m_1 \times m_2 + m_3 + m_1)$	63.77	61.83	43.57

Table 10
The experiment on the incremental of new data using 32-bit.

Number of New Data	MAP@10	Precision@500	Time (s)
9000	47.97	46.14	332.25
19,000	56.77	52.38	281
29,000	61.14	56.52	284.49
39,000	64.28	57.88	317.26
49,000	64.36	58.92	320.78
59,000	64.47	58.93	318.34

number of additional enhancement nodes in incremental step m_3 are set as 10, 200, and 1000, respectively.

Table 8 shows retrieval results of the experiment of incremental of enhancement layer. Each row of the table shows the retrieval performance after each addition of m_3 enhancement nodes. According to Table 8, MAP@50 and Precision@500 improve steadily when BH increases enhancement nodes gradually. Moreover, training times of incremental steps are shorter than that of the initial training. It is because the BH network only trains new network nodes instead of the full network.

The incremental BH network of the second experiment extracts more features through adding m_1 shallow-mapped nodes in iteration. For each incremental step, m_1 shallow-mapped nodes and $m_1 \times m_2$ enhancement nodes are added to the BH network. Moreover, additional m_3 enhancement nodes are added to the BH network (same as the first experiment). Retrieval results are presented in Table 9. Each row of the table shows the retrieval performance after each incremental step. Compared with Table 8, BH yields higher and more stable MAP@50 performance in each step. It shows that adding shallow-mapped nodes to extract more shallow features improves retrieval performance when adding enhancement nodes only does not provide satisfactory improvement of performance.

Table 10 shows the retrieval performance of BH with the increment of data. MAP@10 and Precision@500 show that the retrieval performance of BH network improves with the addition of new data. On the other hand, the training time of each incremental of 10,000 data is less than the initial training time with 9000 data. These show that the incremental BH is effective and efficient.

5. Conclusion

In this paper, we propose a novel broad hashing (BH) method for image retrieval. BH utilizes broad hashing network to learn compact hash codes with the Hadamard code table as a premise. BH yields better retrieval performance compared with most of conventional supervised methods and greatly improves training efficiency compared with deep hashing methods. BH does not lose precision on short hash codes (i.e., 16-bit) while yields high efficiency on long hash codes (i.e., 256-bit). Moreover, three incremental algorithms for BH are developed for fast incremental updates without retraining with the whole dataset. Experimental results on three datasets show the effectiveness of proposed methods.

In BH, number of nodes to be added are set manually, which are not obtained by adaptive learning. Thus, an adaptive method for automatic selection of network nodes will be considered in our future works. When new data arrives, the number of nodes in the BH network will be adjusted adaptively according to the retrieval performance and a new network structure which is more suitable for the current data distribution will be built. Furthermore, new classes will appear in dynamic data environments. Whether broad hashing can be used to study the emergence of new classes is also a meaningful future research topic.

CRediT authorship contribution statement

Wing W.Y. Ng: Conceptualization, Funding acquisition, Supervision, Writing – review & editing. **Xuyu Liu:** Conceptualization, Methodology, Validation, Writing – original draft. **Xing Tian:** Data curation, Validation, Visualization, Writing – review & editing. **Ting Wang:** Methodology, Software, Validation, Conceptualization, Writing – review & editing. **Jianjun Zhang:** Methodology, Software, Validation, Visualization, Writing – review & editing. **C.L. Philip Chen:** Methodology, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Xing Tian reports financial support was provided by National Natural Science Foundation of China. Wing W. Y. Ng reports financial support was provided by National Natural Science Foundation of China. Ting Wang reports financial support was provided by National Natural Science Foundation of China. Jianjun Zhang reports financial support was provided by National Natural Science Foundation of China. Wing W. Y. Ng reports financial support was provided by Guangdong Basic and Applied Basic Research Foundation. Ting Wang reports financial support was provided by Guangdong Basic and Applied Basic Research Foundation. Jianjun Zhang reports financial support was provided by Guangdong Basic and Applied Basic Research Foundation.

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant 62202175, 62476100, 62406115 and 62402186 and in part by Guangdong Basic and Applied Basic Research Foundation (2024A1515011896, 2023A1515012943, 2022A15 15110568, and 2022A1515110112).

Data availability

Public datasets are used.

References

[1] D.V.S. Kishore, C. Phani Kumar, Analysis on content based image retrieval - a logical survey, in: 2022 International Mobile and Embedded Technology Conference, MECON, 2022.

[2] M. Datar, N. Immorlica, P. Indyk, V.S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: Proceedings of the Twentieth Annual Symposium on Computational Geometry, SCG '04, 2004, pp. 253–262.

[3] X. Li, C. Ma, J. Yang, X. Huang, Discrete locally-linear preserving hashing, in: 2018 25th IEEE International Conference on Image Processing, ICIP, 2018, pp. 490–494, http://dx.doi.org/10.1109/ICIP.2018.8451183.

[4] B. Dai, R. Guo, S. Kumar, N. He, L. Song, Stochastic generative hashing, in: D. Precup, Y.W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 70, PMLR, 2017, pp. 913–922.

[5] Y. Yang, Q. Li, X. Tian, W.W.Y. Ng, H. Wang, J. Kittler, M. Gales, R. Cooper, Unsupervised multi-hashing for image retrieval in non-stationary environments, in: 2023 15th International Conference on Advanced Computational Intelligence, ICACI, 2023, pp. 1–6, http://dx.doi.org/10.1109/ICACI58115.2023.10146177.

[6] Y. Chen, Z. Tian, H. Zhang, J. Wang, D. Zhang, Strongly constrained discrete hashing, IEEE Trans. Image Process. 29 (2020) 3596–3611, http://dx.doi.org/10.1109/TIP.2020.2963952.

[7] A. Oliva, A. Torralba, Modeling the shape of the scene: A holistic representation of the spatial envelope, Int. J. Comput. Vis. 42 (3) (2001) 145–175, http://dx.doi.org/10.1023/A:1011139631724.

[8] A. Krizhevsky, I. Sutskever, G.E. Hinton, ImageNet classification with deep convolutional neural networks, Commun. ACM 60 (6) (2017) 84–90, http://dx.doi.org/10.1145/3065386.

[9] R. Xia, Y. Pan, H. Lai, C. Liu, S. Yan, Supervised hashing for image retrieval via image representation learning, in: Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, AAAI '14, AAAI Press, 2014, pp. 2156–2162.

[10] H. Lai, Y. Pan, Y. Liu, S. Yan, Simultaneous feature learning and hash coding with deep neural networks, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, pp. 3270–3278, http://dx.doi.org/10.1109/CVPR.2015.7298947.

[11] Q. Li, Z. Sun, R. He, T. Tan, Deep supervised discrete hashing, in: Advances in Neural Information Processing Systems, Vol. 30, 2017.

[12] F. Cakir, K. He, S.A. Bargal, S. Sclaroff, Hashing with mutual information, IEEE Trans. Pattern Anal. Mach. Intell. 41 (10) (2019) 2424–2437, http://dx.doi.org/10.1109/TPAMI.2019.2914897.

[13] K. Lin, J. Lu, C.-S. Chen, J. Zhou, Learning compact binary descriptors with unsupervised deep neural networks, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 1183–1192, http://dx.doi.org/10.1109/CVPR.2016.133.

[14] T.-T. Do, A.-D. Doan, N.-M. Cheung, Learning to hash with binary deep neural network, in: B. Leibe, J. Matas, N. Sebe, M. Welling (Eds.), Computer Vision – ECCV 2016, Cham, 2016, pp. 219–234.

[15] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, H.T. Shen, Unsupervised deep hashing with similarity-adaptive and discrete optimization, IEEE Trans. Pattern Anal. Mach. Intell. 40 (12) (2018) 3034–3044, http://dx.doi.org/10.1109/TPAMI.2018.2789887.

[16] Z. Ni, Z. Ji, L. Lan, Y.-H. Yuan, X. Shen, Unsupervised discriminative deep hashing with locality and globality preservation, IEEE Signal Process. Lett. 28 (2021) 518–522, http://dx.doi.org/10.1109/LSP.2021.3059526.

[17] L. Jin, Z. Li, J. Tang, Deep semantic multimodal hashing network for scalable image-text and video-text retrievals, IEEE Trans. Neural Netw. Learn. Syst. 34 (4) (2023) 1838–1851, http://dx.doi.org/10.1109/TNNLS.2020.2997020.

[18] Y. Yang, Q. Li, X. Tian, W.W.Y. Ng, H. Wang, J. Kittler, M. Gales, R. Cooper, Unsupervised multi-hashing for image retrieval in non-stationary environments, in: 2023 15th International Conference on Advanced Computational Intelligence, ICACI, 2023, pp. 1–6, http://dx.doi.org/10.1109/ICACI58115.2023.10146177.

[19] C.L.P. Chen, Z. Liu, Broad learning system: An effective and efficient incremental learning system without the need for deep architecture, IEEE Trans. Neural Netw. Learn. Syst. 29 (1) (2018) 10–24, http://dx.doi.org/10.1109/TNNLS.2017.2716952.

[20] V.E. Liong, J. Lu, L.-Y. Duan, Y.-P. Tan, Deep variational and structural hashing, IEEE Trans. Pattern Anal. Mach. Intell. 42 (3) (2020) 580–595, http://dx.doi.org/10.1109/TPAMI.2018.2882816.

[21] Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 35 (12) (2013) 2916–2929, http://dx.doi.org/10.1109/TPAMI.2012.193.

[22] F. Shen, C. Shen, W. Liu, H.T. Shen, Supervised discrete hashing, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2015, pp. 37–45, http://dx.doi.org/10.1109/CVPR.2015.7298598.

[23] Y. Xiao, W. Zhang, X. Dai, X. Dai, N. Zhang, Robust supervised discrete hashing, Neurocomputing 483 (2022) 398–410, http://dx.doi.org/10.1016/j.neucom.2021.09.077.

[24] X. Liu, X. Nie, Q. Zhou, X. Xi, L. Zhu, Y. Yin, Supervised short-length hashing, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI '19, AAAI Press, 2019, pp. 3031–3037.

[25] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, S.-F. Chang, Supervised hashing with kernels, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 2074–2081, http://dx.doi.org/10.1109/CVPR.2012.6247912.

[26] W.-C. Kang, W.-J. Li, Z.-H. Zhou, Column Sampling Based Discrete Supervised Hashing, AAAI '16, AAAI Press, 2016, pp. 1230–1236.

[27] Y. Shi, X. Nie, X. Liu, L. Zou, Y. Yin, Supervised adaptive similarity matrix hashing, IEEE Trans. Image Process. 31 (2022) 2755–2766, http://dx.doi.org/10.1109/TIP.2022.3158092.

[28] Z. Zhang, X. Zhu, G. Lu, Y. Zhang, Probability ordinal-preserving semantic hashing for large-scale image retrieval, ACM Trans. Knowl. Discov. Data 15 (3) (2021) URL https://doi.org/10.1145/3442204.

[29] Y. Sun, J. Dai, Z. Ren, Q. Li, D. Peng, Relaxed energy preserving hashing for image retrieval, IEEE Trans. Intell. Transp. Syst. 25 (7) (2024) 7388–7400, http://dx.doi.org/10.1109/TITS.2024.3351841.

[30] X. Nie, X. Liu, J. Guo, L. Wang, Y. Yin, Supervised discrete multiple-length hashing for image retrieval, IEEE Trans. Big Data 9 (1) (2023) 312–327, http://dx.doi.org/10.1109/TBDATA.2022.3161905.

[31] Y. Sun, X. Wang, D. Peng, Z. Ren, X. Shen, Hierarchical hashing learning for image set classification, IEEE Trans. Image Process. 32 (2023) 1732–1744, http://dx.doi.org/10.1109/TIP.2023.3251025.

[32] Q. Ma, C. Bai, J. Zhang, Z. Liu, S. Chen, Supervised learning based discrete hashing for image retrieval, Pattern Recognit. 92 (C) (2019) 156–164, http://dx.doi.org/10.1016/j.patcog.2019.03.022.

[33] X. Zhe, S. Chen, H. Yan, Deep class-wise hashing: Semantics-preserving hashing via class-wise loss, IEEE Trans. Neural Netw. Learn. Syst. 31 (5) (2020) 1681–1695, http://dx.doi.org/10.1109/TNNLS.2019.2921805.

[34] A. Jose, D. Filbert, C. Rohlfing, J.-R. Ohm, Deep hashing with hash center update for efficient image retrieval, in: ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, 2022, pp. 4773–4777, http://dx.doi.org/10.1109/ICASSP43922.2022.9746805.

[35] X. Zheng, Y. Zhang, X. Lu, Deep balanced discrete hashing for image retrieval, Neurocomputing 403 (2020) 224–236, http://dx.doi.org/10.1016/j.neucom.2020.04.037.

[36] W. Hu, L. Wu, M. Jian, Y. Chen, H. Yu, Cosine metric supervised deep hashing with balanced similarity, Neurocomputing 448 (2021) 94–105, http://dx.doi.org/10.1016/j.neucom.2021.03.093.

[37] Q.-Y. Jiang, X. Cui, W.-J. Li, Deep discrete supervised hashing, IEEE Trans. Image Process. 27 (12) (2018) 5996–6009, http://dx.doi.org/10.1109/TIP.2018.2864894.

[38] X. Lu, Y. Chen, X. Li, Siamese dilated inception hashing with intra-group correlation enhancement for image retrieval, IEEE Trans. Neural Netw. Learn. Syst. 31 (8) (2020) 3032–3046, http://dx.doi.org/10.1109/TNNLS.2019.2935118.

[39] T. Li, Z. Zhang, L. Pei, Y. Gan, HashFormer: Vision transformer based deep hashing for image retrieval, IEEE Signal Process. Lett. 29 (2022) 827–831, http://dx.doi.org/10.1109/LSP.2022.3157517.

[40] S.R. Dubey, S.K. Singh, W.-T. Chu, Vision transformer hashing for image retrieval, in: 2022 IEEE International Conference on Multimedia and Expo, ICME, 2022, pp. 1–6, http://dx.doi.org/10.1109/ICME52920.2022.9859900.

[41] Z.-D. Chen, X. Luo, Y. Wang, S. Guo, X.-S. Xu, Fine-grained hashing with double filtering, IEEE Trans. Image Process. 31 (2022) 1671–1683, http://dx.doi.org/10.1109/TIP.2022.3145159.

[42] X. Gong, T. Zhang, C.L.P. Chen, Z. Liu, Research review for broad learning system: Algorithms, theory, and applications, IEEE Trans. Cybern. (2021) 1–29, http://dx.doi.org/10.1109/TCYB.2021.3061094.

[43] F. Chu, T. Liang, C.L.P. Chen, X. Wang, X. Ma, Weighted broad learning system and its application in nonlinear industrial process modeling, IEEE Trans. Neural Netw. Learn. Syst. 31 (8) (2020) 3017–3031, http://dx.doi.org/10.1109/TNNLS.2019.2935033.

[44] S. Feng, C.P. Chen, Fuzzy broad learning system: A novel neuro-fuzzy model for regression and classification, IEEE Trans. Cybern. 50 (2) (2020) 414–424, http://dx.doi.org/10.1109/TCYB.2018.2857815.

[45] R. Mao, R. Cui, C.L.P. Chen, Broad learning with reinforcement learning signal feedback: Theory and applications, IEEE Trans. Neural Netw. Learn. Syst. 33 (7) (2022) 2952–2964, http://dx.doi.org/10.1109/TNNLS.2020.3047941.

[46] G. Liu, W. Shen, L. Gao, A. Kusiak, Predictive modeling with an adaptive unsupervised broad transfer algorithm, IEEE Trans. Instrum. Meas. 70 (2021) 1–12, http://dx.doi.org/10.1109/TIM.2021.3088496.

[47] Y. Kong, Y. Cheng, C.L.P. Chen, X. Wang, Hyperspectral image clustering based on unsupervised broad learning, IEEE Geosci. Remote. Sens. Lett. 16 (11) (2019) 1741–1745, http://dx.doi.org/10.1109/LGRS.2019.2907598.

[48] H. Zhao, J. Zheng, W. Deng, Y. Song, Semi-supervised broad learning system based on manifold regularization and broad network, IEEE Trans. Circuits Syst. I. Regul. Pap. 67 (3) (2020) 983–994, http://dx.doi.org/10.1109/TCSI.2019.2959886.

[49] Z. Xuan, C.-E. Ren, Z. Shi, Y. Guan, Semi-supervised domain adaption classifier via broad learning system, in: 2021 IEEE International Conference on Systems, Man, and Cybernetics, SMC, 2021, pp. 2743–2748, http://dx.doi.org/10.1109/SMC52423.2021.9658809.

[50] F. Hao, J. Jin, A broad learning ensemble system using bagging for typhoon trajectory forecasting, in: 2022 2nd International Conference on Consumer Electronics and Computer Engineering, ICCECE, 2022, pp. 729–733, http://dx.doi.org/10.1109/ICCECE54139.2022.9712832.

[51] X. Peng, K. Ota, M. Dong, Edge computing based traffic analysis system using broad learning, in: Artificial Intelligence for Communications and Networks, 2019, pp. 238–251.

[52] A.S. Hedayat, W.D. Wallis, Hadamard matrices and their applications, Ann. Statist. 6 (6) (1978) 1184–1238.

[53] M. Lin, R. Ji, H. Liu, X. Sun, S. Chen, Q. Tian, Hadamard matrix guided online hashing, Int. J. Comput. Vis. 128 (8–9) (2020) 2279–2306.
[54] J. Gui, T. Liu, Z. Sun, D. Tao, T. Tan, Fast supervised discrete hashing, IEEE Trans. Pattern Anal. Mach. Intell. 40 (2) (2018) 490–496, http://dx.doi.org/10.1109/TPAMI.2017.2678475.
[55] X. Xu, Z. Lai, Y. Chen, Relaxed locality preserving supervised discrete hashing, IEEE Trans. Big Data 8 (4) (2022) 1118–1128, http://dx.doi.org/10.1109/TBDATA.2020.3027379.
[56] X. Lu, Y. Chen, X. Li, Hierarchical recurrent neural hashing for image retrieval with hierarchical convolutional features, IEEE Trans. Image Process. 27 (1) (2018) 106–120, http://dx.doi.org/10.1109/TIP.2017.2755766.
[57] Y. Cao, M. Long, B. Liu, J. Wang, Deep Cauchy hashing for hamming space retrieval, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2018, pp. 1229–1237, http://dx.doi.org/10.1109/CVPR.2018.00134.
[58] L. Fan, K.W. Ng, C. Ju, T. Zhang, C.S. Chan, Deep polarized network for supervised learning of accurate binary hashing codes, in: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, 2020, pp. 825–831, http://dx.doi.org/10.24963/ijcai.2020/115.

Dr. Wing W. Y. Ng received the B.Sc. and Ph.D. degrees from Hong Kong Polytechnic University, Hong Kong, in 2001 and 2006, respectively. He is a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China, where he also is the Deputy Director of the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information. His current research interests include neural networks, deep learning, smart grid, smart healthcare, smart manufacturing, and nonstationary information retrieval. Dr. Ng is currently an Associate Editor of the International Journal of Machine Learning and Cybernetics. He is the Principal Investigator of five China National Natural Science Foundation projects, a Program for New Century Excellent Talents in University from China Ministry of Education, and several provincial-level projects. He served as the Board of Governor for IEEE Systems, Man and Cybernetics Society in 2011 and 2013.

Xuyu Liu received the B.E. degree in Computer Science and Technology from China Agricultural University and the M.E. degree in the same field from South China University of Technology. Her research interests include image retrieval, object tracking, and deep learning.

Dr. Xing Tian received the B.Sc. and Ph.D. degrees from the South China University of Technology, in 2014 and 2019, respectively. He is an associate professor with the School of Artificial Intelligence, South China Normal University, Guangzhou, China. He was a postdoctoral researcher with the South China University of Technology, and with City University of Hong Kong, in 2019 and 2020, respectively. He has also visited the Ulster University in U.K for one year as an academic visitor supported by China Scholarships Council (CSC). His major research interests include machine learning, hashing, image retrieval, cross-modal retrieval, and deep learning.

Ting Wang (Member, IEEE) received the B.Sc. degree from the School of Computer Science and Technology, Guizhou University, Guiyang, China, in 2014, the M.Sc. degree in computer science from Northeast Normal University, Changchun, China, in 2017, and the Ph.D. degree from the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China, in 2021. She was a Post-Doctoral Research Fellow with the Department of Radiology, Guangzhou First People's Hospital, School of Medicine, South China University of Technology, Guangzhou, China, from July 2021 to July 2024. She is currently an Associate Professor with the College of Electronic Engineering, College of Artificial Intelligence, South China Agricultural University, Guangzhou, China. Her current research interests include learning methods and generalization capabilities for deep neural networks and broad networks, and their applications in real-world problems, such as image retrieval, medical image, and smart grid.

Jianjun Zhang (Member, IEEE) received the bachelor's and Ph.D. degrees in computer science from the South China University of Technology, Guangzhou, China, in 2015 and 2020, respectively. He was a Post-Doctoral Research Fellow at the School of Computer Science and Engineering, South China University of Technology, Guangzhou, China, from Jan 2021 to Jan 2024. He is currently an Associate Professor with College of Mathematics and Informatics, South China Agricultural University, Guangzhou, China. Dr. Zhang was awarded the Best Paper Award at the IEEE ICCI*CC'22. He has published over 40 academic papers on reputable journals and conferences, including IEEE Transactions on Neural Networks and Learning Systems and IEEE Transactions on Cybernetics. His current research focuses on incremental learning under nonstationary and imbalanced environments.

C. L. Philip Chen (Fellow, IEEE) received the M.S. degree in electrical engineering from the University of Michigan, Ann Arbor, MI, USA, in 1985, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1988. He is the Chair Professor and the Dean with the College of Computer Science and Engineering, South China University of Technology, Guangzhou, China. His current research interests include cybernetics, systems, and computational intelligence. Dr. Chen received the IEEE Norbert Wiener Award in 2018 for his contribution in Systems and Cybernetics, and Machine Learnings, the IEEE Joseph G. Wohl Outstanding Career Award, and received two times Best Transactions Paper Award from the IEEE Transactions on Neural Networks and Learning Systems for his papers in 2014 and 2018. He was a recipient of the 2016 Outstanding Electrical and Computer Engineers Award from his alma mater, Purdue University in 1988, after he graduated from the University of Michigan, Ann Arbor, MI, USA in 1985. He is a Highly Cited Researcher by Clarivate Analytics from 2018 to 2023. He was the Editor-in-Chief of IEEE Transactions on Cybernetics and IEEE Transactions on Systems, Man, and Cybernetics: Systems, and the President of IEEE Systems, Man, and Cybernetics Society. He is a member of Academia Europaea and European Academy of Sciences and Arts. He is a Fellow of AAAS, IAPR, CAA, CAAI, and HKIE.