Multi-Level Supervised Hashing with Deep Features for Efficient Image Retrieval

[1]Wing W. Y. Ng, [1]Jiayong Li, [1,2]Xing Tian*, [3]Hui Wang, [2]Sam Kwong, [3]Jonathan Wallace

[1]*Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, Computer Science and Engineering, South China University of Technology, Guangzhou, China*
[2]*Department of Computer Science, City University of Hong Kong, Hong Kong, China*
[3]*School of Computing, Ulster University, Jordanstown, United Kingdom*
**Corresponding Author*

Abstract

Image hashing based on deep convolutional neural networks (CNN), deep hashing, has acquired breakthrough in image retrieval. Although deep features from various CNN layers have various levels of information, most of the existing deep hashing methods extract the feature vector only from the output of the penultimate fully-connected layer, focusing primarily on semantic information whilst ignoring detailed structure information. This calls for research on multi-level hashing, utilizing multi-level features to exploit different levels of CNN characteristics. To fill this gap, a novel image hashing method, Multi-Level Supervised Hashing with deep feature (MLSH), is proposed in this paper to further exploit multiple levels of deep image features. It uses a multiple-hash-table mechanism to integrate multi-level features extracted from an individual deep convolutional neural network. It takes advantage of the complementarity among multi-level features from various layers of a single deep network. High-level features reveal the semantic content of the image, while low-level features provide the structural information that is missing in high-level features. Instead of simple concatenation, several hash tables are trained individually using different levels of features from different layers, which are then integrated for efficient image retrieval. The method has been systematically evaluated through experiments on three image databases, including CIFAR-10, MNIST and NUSWIDE, and

has thus been demonstrated to set a new state of the art in image hashing, outperforming several state-of-the-art hashing methods. Furthermore, the recall and precision can be balanced and improved simultaneously.

Keywords: multi-table mechanism, multi-level deep feature, image retrieval, structural and semantic similarity

1. INTRODUCTION

Hashing [1, 2, 3, 4] is a key enabling technique for ANN based image retrieval. For hashing-based methods, high-dimensional feature vectors are converted into low-dimension binary codes (hash codes), and the Hamming distance is calculated between hash codes as surrogate for the distance between images. Given a query, the image with the least Hamming distance to the query is then returned.

Traditional hashing-based methods [1, 2, 3] have already achieved good retrieval performance, but they have not advanced much in terms of performance in recent years because of the limitations of hand-crafted features they have used. Most hashing methods employs hand-crafted features, including Histogram of Oriented Gradients (HOG) [5], Local Binary Pattern (LBP) [6], Scale Invariant Feature Transform (SIFT) [7] and GIST [8], which are unsupervised and thus cannot fully capture the underlying semantic similarity of images. Recently, instead of hand-crafted features, deep features from deep neural networks are being used as image descriptors to overcome the semantic gap. Deep features are more informative and more relevant than hand crafted features, resulting in moderate improvement in retrieval performance [9].

Normally, a deeper convolutional layer extracts a more abstract feature map/vector, resulting in a higher-level feature. Higher-level features capture richer semantic information but lack finer-grained details; lower-level features have higher spatial resolution but suffer more with background cluttering and semantic blurring. Recent works on deep hashing have extracted the penultimate layer features from a neural network [10, 11] as the global image descriptor. They are however not desirable image features due to high-level features hav-

ing rich semantic information but lacking spatial resolution. To alleviate this problem, multi-level features, which are feature vectors extracted from various network layers, can be used to consider both semantic and structure information. To calculate similarity, images must first of all be represented in terms of features, which may be extracted using manually designed feature descriptors or, more recently, using deep neural networks such as convolutional neural network (CNN). Each layer of a CNN may be used as a feature vector for a given image, so multiple feature vectors are available to represent one image. In terms of its semantics, features from various CNN layers have different characteristics, thus they complement each other. How to utilize different levels of features, in particular, how to exploit their complementarity for more effective image hashing is a challenge. To the best of our knowledge, this challenge has not been researched in the literature. A related area of research is multi-view hashing [12, 13, 14, 15], where different types of hand-crafted descriptors are fused together by concatenation. Sequential Spectral Learning to Hash [14] determines the best averaged distance matrix by minimizing -averaging view-specific distance matrices. Multi-view anchor graph hashing (MVAGH) [12] computes the subset of eigenvectors of the average similarity matrix to non-linearly combine binary code. Deep Multimodal Hashing with Orthogonal Regularization (DMHOR) [15] merges various deep features and performs similarity search on multimodal features. Multimodal features are extracted from different network structure. However, these methods mainly employ hand-crafted features or multimodal features, which do not exploit the relationship between different views of features.

To exploit multi-level deep features more thoroughly, this paper proposes a novel image deep hashing method, *Multi-Level Supervised Hashing with deep feature*, to take advantage of the complementarity of multi-level features extracted from various layers of a deep neural network. Instead of directly concatenating feature vectors from multiple layers, a multiple-hash-table mechanism is proposed to make better use of multi-level features. One hash table is trained using one level of deep features, and then all hash tables are integrated. It

returns the intersection set of several returned image sets corresponding to multiple hash tables. The direct concatenation manner is not an efficient way to exploit the high-level semantic similarity and the low-level fine-grained differences. With multiple hash tables being used, high-level features can guarantee good semantic preservation, while low-level features can avoid the influence of high-level features so as to impose a vital role in differentiating fine-grained distinctions. Moreover, the use of a multi-table mechanism [16] ensures that the retrieval recall can be improved while preventing precision degradation.

To the best of our knowledge, this is the first study of multi-level deep hashing where complementarity between different levels of features is exploited. The proposed method can train hash codes preserving both high-level semantic similarity and low-level structural similarity. A related area of research is multi-view hashing, where different feature vectors are extracted from the same image, which are then concatenated into a single one. Different features vector comes from different hand-crafted features (e.g. SIFT, LBP and HOG), or features extracted from different scales. The relationship between different views is unclear and is not exploited.

The main contributions of this paper are as follows:

- A novel hashing method, *Multi-Level Supervised Hashing with deep feature*, is proposed. Multiple levels of deep features are extracted from different layers of the deep convolutional neural network to better preserve semantic and structural information simultaneously. The use of a multi-level mechanism ensures the generated hash codes are more discriminative and informative.

- Multi-level features are combined by training separate hash tables respectively, which are then integrated. The integration of several hash tables can better take advantage of different-level features by capturing both semantic and structural information in images. Moreover, recall and precision can be balanced for better retrieval performance.

- Three widely-used image databases are employed to evaluate the proposed

method. Experimental results show that the proposed method outperforms other state-of-the-art hashing methods.

This paper is organized as follows. Related works are described in Section 2 and the proposed method is detailed in Section 3. The experimental results and the conclusion of the paper are shown in Section 4 and Section 5, respectively.

2. Related Works

Content-based image retrieval (CBIR) has made substantial advancement recently. It aims to return a collection of images similar to the query based on the content rather than the metadata of images. One approach to solving this problem is exhaustive search, which is impractical for large-scale databases. Traditional CBIR methods first extract image features and then return similar images based on the distance of image feature vectors using e.g. cosine distance and Euclidean distance. However, they are impractical for real-world databases due to the high computational cost. In the case of a large-scale database or a high dimensionality of features, the cost of finding an exact accurate nearest image is very high. An alternative approach is *Approximate Nearest Neighbor* (ANN), which trades off retrieval accuracy for speed. Hashing methods are proven to be efficient as one of the ANN for image retrieval. According to whether the image label information is used, hashing can be generally divided into three categories: unsupervised, semi-supervised and supervised.

For the first category, one of the most commonly-used techniques is Locality Sensitive Hashing (LSH) [1], which randomly generates projections to map data from high dimensional real-value space to low dimensional Hamming space. Locality-sensitive binary codes from shift-variant kernels (SKLSH) [2] and kernelized LSH [3] are the improvements of LSH, which assume the data is multidimensional and utilize a kernel function to discover the underlying structure of the images. However, they are data-independent as they ignore the semantic structure of the data. Iterative Quantization (ITQ) [17] is a data-dependent method, which minimizes the quantization loss to binarize the outcomes. Spec-

tral Hashing (SH) [18] is presented to generate balanced hash codes via minimizing the correlations among different hash functions. Asymmetric Cyclical Hashing (ACH) [19] uses shorter hash codes for database images and longer hash codes for query images to reduce storage cost. Semi-supervised hashing utilizes only a small part of image labels. Semi-supervised hashing (SSH) [20] aims to generate balanced hash codes and avoid over-fitting by minimizing the empirical errors between pairwise data. Semi-supervised nonlinear hashing using bootstrap sequential projection learning (BSPLH) [21] sequentially trains several hash functions by amending the errors of all previous hash functions.

Supervised hashing requires the whole database to be labeled. Inspired by latent structural SVM, Minimum Loss Hash (MLH) [22] is proposed to use structural SVMs with latent variables and the hinge loss function to generate hash codes. Kernel-based Supervised Hashing (KSH) [23] applies the kernel function to generate the similarity-preserving codes. Binary Reconstructive Embedding (BRE) [24] explicitly minimizes the construction error between data in the original space and that in the Hamming space. Column Sampling based Discrete Supervised Hashing (CODISH) [25] learns discrete hash codes directly in a supervised manner.

In addition to the traditional hashing using hand-crafted features, some deep hashing methods are presented recently to exploit the strong representation capability of the deep neural network. Conventional Neural Network Hashing (CNNH) [26] learns the hash function and the feature representation independently, where the hash function learning cannot feedback to the feature learning. To address this problem, Lai et al. [27] uses a ranking loss based on a triplet of images to jointly learn the hash codes and feature representation so that the hash function learning can provide feedback to the feature learning. Deep Semantic Preserving and Ranking-based Hashing (DSRH) [28] is also a triplet-based deep hashing which reduces bit redundancy with the orthogonal constraints. Deep Supervised Hashing (DSH) [29] is a pairwise-based hashing to generate the discriminative hash codes. Discriminative Deep Hashing (DDH) [30] presents a divide-and-encode module to maximize the discriminability of

the hash codes. Different bits contribute differently to the image retrieval, so they should be treated differently. Bit-Scalable Deep Hashing (DRSCH) [31] uses a weighting scheme to generate the compact and bit-scalable hash codes. Zhang et al. [32] applies the query-adaptive scheme to provide more accurate ranking. WMRDH [33] presents a order-aware ranking loss with a weighting scheme to generate similarity preserving hash codes. In addition, studies on cross-modal hashing are also an important research filed. [34] proposes to learn hash functions in a domain-adaptive limited text space (DALTS), instead of image space and common space. [35] applies multitask learning with Capped-1 penalty to map images to semantic concept probability distribution. And then a knowledge-based concept transferring algorithm is applied for better global semantic capturing.Table 1 demonstrates the comparison between the proposed method and representative aforementioned hashing methods.

Because the aforementioned methods are based on a single table, the precision drops when the desired number of similar images increases. Multi-table hashing is proposed to trade off the recall and precision. Semi-supervised nonlinear hashing using bootstrap sequential projection learning (BSPLH) [21] sequentially trains several hash functions in a boosting manner. WMRDH [33] also weighted integrates several hash tables which are trained with the proposed order-aware ranking loss function. However, most of the multi-table hashing methods are based on the same single-view features, which cannot better capture the image information.

The aforementioned methods apply only one type of feature. However, different types of feature contain different information. Inspired by the multi-level feature aggregation scheme [11, 36], multiple different types of features should be merged to make a more comprehensive descriptor. Several multi-level image retrieval methods are proposed, which can be roughly classified into three groups. The first one [12, 13] is to cut out several regions from an image or transform an image into different resolutions, and then extract features from different regions or resolutions of an image. Multi-view Anchor Graph Hashing (MVAGH) [12] seeks to find a non-linear combination of binary codes. Compos-

Table 1: Comparison between the proposed method and the state-of-the-art methods.

		Method	Differences
Traditional Hashing	Supervised	BRE [24]	1. BRE applies the single-view hand-crafted features, while MLSH applies multi-level deep CNN features. 2. BRE aims to minimize the construction error between data in the original space and that in the Hamming space, while MLSH solves the objective function based on column-sample space.
		KSH [23]	KSH applies the kernel function, while MLSH extracts deep features and trains hash function in original deep feature space.
		MLH [22]	MLH trains hash functions using structural SVMs. MLSH aims to minimize quantization error.
	Unsupervised	ITQ [17]	MLSH applied the sample-column space to improve the efficiency. ITQ solves the discrete problem directly.
		SH [18]	SH focuses on reducing the correlation between hash functions. MLSH takes it as one of the factors in the objective function, and it also considers the quantization error.
		LSH [1]	LSH is data-independent, which generates hash codes randomly. MLSH is data-dependent and it learns hash functions based on its data distribution.
Deep Hashing	Supervised	DDH [30]	1.MLSH trains hash function after extracting different levels CNN features. All the comparison methods only apply the high-level CNN features. 2.NINH, DSRH and DRSCH are based on triplet loss, and DSH is based on pairwise loss. The objective function of MLSH is based on the image itself. 3.All of the comparison methods only consider the high level semantic similarity. MLSH also consider the low-level structural information.
		DSH [29]	
		DRSCH [31]	
		DSRH [28]	
		NINH [27]	
		CNNH [26]	

ite Hash (CHMIS) with multiple information sources [13] applies a weighting scheme to multiple feature sources to maximize the coding performance. [37] emposes a geometric-constrained, and calculates the cost value of the energy

function from multiple images, which is aggregated in an image space using a semi-global optimization approach. The fine-grained spatial-temporal attention model (FSTA) [38] uses a two layer LSTM with the attention mechanism on the frame level to consider the spatial-region information. Most existing multi-level hashing methods [12, 13] belong to the first category. The second one [39, 40, 41, 42] is early fusion, which encodes different types of descriptors into a single feature vector. In [42], hierarchical recurrent neural network is exploited to construct pyramid image representation and generate a single effective hash table. However, the direct combination of different descriptors reduces the effectiveness of various features. The last one is late fusion [43], which combine the retrieval results of different descriptors. In [40], CNN features are extracted from various layers which are then encoded as a single feature vector by vector of locally aggregated descriptors (VLAD) encoding. It measures the similarity based on Euclidean distance which is time consuming.

The aforementioned works are in the field of feature extraction. Several works on feature selection have also been studied in recent years. [44] presents a hybrid of particle swarm optimization algorithm with genetic operators to select features. [45] presents to use krill herd algorithms to generate the efficient subset for text clustering. Both of the previous two methods use k-means as the evaluation.

3. Multi-Level Supervised Hashing with Deep Feature

Normally, a convolutional neural network contains several convolutional blocks, each of which is constructed with one or more convolutional layers and pooling layers. The flow diagram of MLSH is demonstrated in Fig. 1. The red star is the query image. Empty circles of different colors represent the returned region with a certain Hamming distance, where the images inside should be returned. The purple circle is the center of the returned region. The red circle is the desired returned region. MLSH divides the whole CNN into Q convolutional blocks. The proposed network applied the classification layer as the last fully-

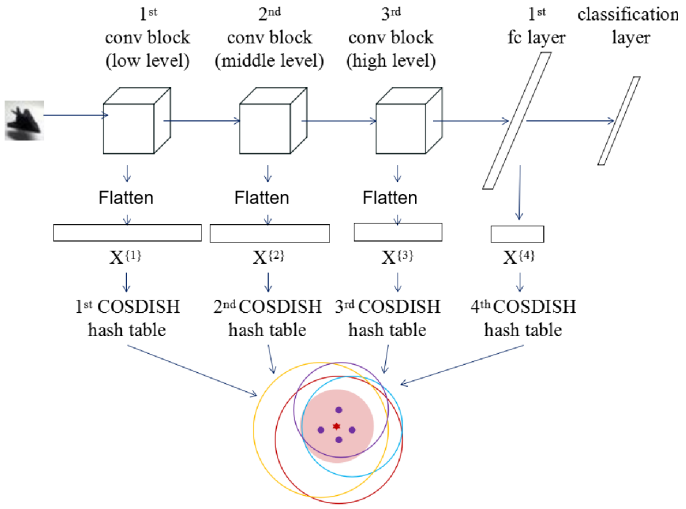


Figure 1: The flow diagram of MLSH. The red star is the query image. Empty circles of different colors represent the returned region with a certain Hamming distance, where the images inside should be returned. The purple circle is the center of the returned region. The red circle is the desired returned region.

connected layer to form a classifier. Features are generated by minimizing the classification loss introduced in section 3.1. Assume that a set of N training images from C classes with labels Y is given, the proposed method first extracts a set of feature matrices $X = \{X^{\{1\}}, ..., X^{\{Q\}}\}$ of various levels from the output of various convolutional blocks and the first fully-connected layer in a deep convolutional network, where Q denotes the number of levels and $X^{\{q\}} \in R^{N \times d^{(q)}}$ denotes the feature matrix of the q^{th} level, and $d^{(q)}$ denotes the feature dimension of the q^{th} level. Then, the proposed method applies a supervised hashing method CODISH [25] introduced in section 3.2 to learn a projection function for each feature matrix: $F^{\{q\}} : X^{\{q\}} \rightarrow B^{\{q\}} \in \{-1, 1\}^{N \times K}$, which encodes an image feature $x^{(q)} \in X^{(q)}$ of the q^{th} level into a K-dimensional binary code $b^{(q)} \in B^{(q)}$ in the Hamming space. Instead of being resized to the same dimension, features of various levels are used to train corresponding hash table directly to fully preserve the feature information. Q hash tables of various levels are integrated for image retrieval. $d^{(q)}$ denotes the feature dimension corresponding

to the q^{th} scale. Images belonging to the intersection of the images returned by Q hash tables are finally returned. The integration of multiple hash tables is introduced in Section 3.3. In Fig. 1, the red star and the pink shaded area denote the query and the true neighbor of the query, respectively. The purple red point and the open circle indicate the center of the bucket and the area (Hamming ball) returned by the corresponding hash table according to certain radius, respectively. In this way, MLSH can utilize the complementarity among multi-level features. Moreover, in order to return a desire number of similar images, the radius of the Hamming ball need to be increased, at the cost of lower precision. Using multiple tables to combine features of different levels can better trade off precision and recall, because it returns the intersection of multiple hamming balls, so that less dissimilar images are returned.

In the remainder of this section, Section 3.1 introduces the feature extraction process. The supervised hashing method is described in Section 3.2. The integration of multiple hash tables and the complexity analysis are described in Section 3.3 and 3.4, respectively.

3.1. Multi-level Feature Extraction

Fig. 2 provides the overview of the entire feature extraction process. The convolutional network consists of five layers, including three convolutional layers and two fully-connected layers. The first convolutional layer applies 32 filters and is followed by the max pooling operation, which forms the first convolutional block. The last two convolutional layers apply 32 and 64 filters, respectively, and are followed by the average pooling operation, which form the second and third convolutional blocks respectively. The size of the convolutional kernel and pooling operation are set as 5×5 with the stride 1 and 3×3 with the stride of 2, respectively. There are 500 units for the first fully-connected layer, which adopts the rectified linear activation function. The last fully-connected layer is the classification layer with C units, where C is the number of categories.

For the single-label classification, the softmax function serves as the activa-

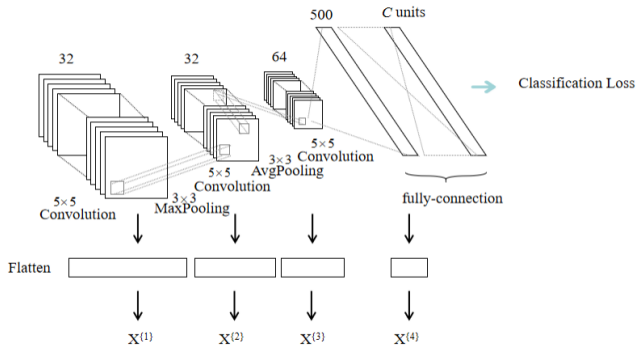


Figure 2: The flow diagram of MLSH.The big rectangle represents the feature map of a convolutional layer and the number over it (e.g. '32') is the number of channel. The size of the rectangle is the width and height of the feature map. The small rectangle is the kernel and the number below it (e.g. '5×5') is the kernel size. The Oblique rectangles following the convolutional layers denotes the fully-connected layers. The rectangles below the convolutional layers are the flatten vectors.

tion function, and the objective function for the CNN training is:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{C} t_{i,j} log(p_{i,j}) \tag{1}$$

where $t_{i,j}$ and $p_{i,j}$ denote the target and prediction of the i^{th} data of j^{th} category. $t_{i,j}$ is 1 if the image i belongs to the j^{th} category, and 0 otherwise.

For the multi-label classification, the sigmoid function serves as the activation function, and the objective function for CNN training is defined as:

$$L = -\sum_{i=1}^{N}\sum_{j=1}^{C}(t_{i,j} \times log(p_{i,j}) + (1 - t_{i,j}) \times log(1 - p_{i,j})) \tag{2}$$

The back-propagation mechanisms are applied to train network parameters. Q levels of feature matrices are extracted from every convolutional block and the penultimate layer of the whole network (i.e. the 1^{st} fc layer in Fig. 1). Assume that the output of a convolutional block consists of C feature maps (each has a height H and a width W), an image is represented by a $H \times W \times C$-dimensional vector. fc_i denotes the i^{th} fully-connected layer and $conv_i$ denotes the i^{th} convolutional layer. In this paper, Q is set to be 4. Thus four feature matrices

are extracted from the first fully-connected layer and three convolutional layers. Feature of each level serves as the input of the supervised hashing method to be introduced in Section 3.2. Therefore, Q hash tables of corresponding levels are generated for image retrieval.

3.2. The Supervised Hashing

The proposed method applies CODISH [25] as the basic supervised hashing method to train the hash table of each individual feature level. The description of CODISH is presented in this section. It is a discrete optimization problem for iteration solutions. It applies the commonly-used objective function, which is defined as:

$$\min_{B\in\{-1,+1\}^{N\times K}} ||KS - BB^T||_F^2 \tag{3}$$

Supposed the training feature matrix is $X \in R^{N\times d}$ and the semantic similarity matrix is $S \in \{-1,+1\}^{N\times N}$, where N is the number of training samples, d is the feature dimension, and $S_{ij} = 1$ denotes data x_i and x_j are semantically similar and $S_{ij} = 0$ otherwise. In each iteration, several data are randomly selected from $X = \{X_1, ..., X_N\}$ to form a subset U^s, and the rest form a subset U^n, where $U^n = X - U^s$. $|U^s|$ and $|U^n|$ denote the number of samples in two data subset respectively. Thus, the sub semantic similarity matrix $\widetilde{S}^{U^s} \in \{-1,+1\}^{|U^s|\times|U^s|}$ between $|U^s|$ selected data and $\widetilde{S}^{U^n} \in \{-1,+1\}^{|U^n|\times|U^s|}$ could be generated. $B^{U^s} \in \{-1,+1\}^{|U^s|\times K}$ and $B^{U^n} \in \{-1,+1\}^{|U^n|\times K}$ represent the generated hash code of the corresponding data, and K is the code length. The objective function Eq. 3 in an iteration can be transformed as:

$$\min_{B^{U^s},B^{U^n}} ||K\widetilde{S}^{U^n} - B^{U^n}(B^{U^s})^T||_F^2 + ||K\widetilde{S}^{U^s} - B^{U^s}(B^{U^s})^T||_F^2 \tag{4}$$

In each iteration, an optimization strategy is applied to solve the objective function. B^{U^s} and B^{U^n} can be iteratively updated by fixing the other.

Update B^{U^n} with B^{U^s} fixed.

When B^{U^s} is fixed, we can convert the objective function into:

$$\min_{B^{U^n}} ||K\widetilde{S}^{U^n} - B^{U^n}(B^{U^s})^T||_F^2 \tag{5}$$

To solve it in a discrete way with a constant-approximation bound, the loss is changed from Frobenius norm to $L1$ norm, which is defined as:

$$\min_{B^{U^n}} ||K\widetilde{S}^{U^n} - B^{U^n}(B^{U^s})^T||_F^2 \tag{6}$$

Obviously, when $B^{U^n} = sgn(\widetilde{S}^{U^n}B^{U^s})$, Eq. 6 reaches its minimum.

In practice, $B_t^{U^n}$ is set as $e\_sgn((\widetilde{S}^{U^n}B^{U^s}, B_{t-1}^{U^n})$ defined in Eq. 7 to avoid the influence of $\widetilde{S}^{U^n}B^{U^s}$ being zero, where t is the iteration index.

$$e\_sgn(a,b) = \begin{cases} 1, & a > 0 \\ b, & a = 0 \\ -1, & a < 0 \end{cases} \tag{7}$$

Update B^{U^n} with B^{U^n} fixed. When B^{U^n} is fixed, the problem can be transformed into K binary quadratic programming (BQP) problems. The optimization of the k^{th} bit of B^{U^n} is defined as

$$\min_{b^k \in \{-1,+1\}^{|U^s|}} (b^k)^T \varphi^{(k)} b^k + (b^k)^T \lambda^{(k)} \tag{8}$$

where

$$\varphi^{(k)}_{\substack{i,j \\ i \neq j}} = -2(K\widetilde{S}^{U^s}_{i,j} - \sum_{m=1}^{k-1} b_i^m b_j^m), \varphi^{(k)}_{i,j} = 0, \tag{9}$$

$$\lambda_i^{(k)} = -2\sum_{l=1}^{|U^n|} B_{l,k}^{|U^n|}(K\widetilde{S}^{U^n}_{l,k} - \sum_{m=1}^{k-1} B^{U^n}_{l,m}B^{U^s}_{i,m} \tag{10}$$

With binary constraint $B \in \{0,1\}$, Eq. 8 is transformed into a standard BQP problem, which is defined as:

$$\min_{\overline{b}^k \in \{0,1\}^{|U^S|}} (\overline{b}^k)^T \overline{\varphi}^{(k)} \overline{b}^k + (\overline{b}^k)^T \overline{\lambda}^{(k)} \tag{11}$$

where $\overline{b}^k = \frac{1}{2}(b^k + 1)$, $\overline{\varphi}^{(k)} = 4\varphi^{(k)}$ and $\overline{\lambda_i}^{(k)} = 2(\lambda_i^{(k)} - \sum l = 1^{|U^S|}(\lambda_{i,l}^{(k)} + \lambda_{l,j}^{(k)}))$.

Moreover, Eq. 11 can be turned into Eq. 12 without linear term, and also an additional constraint is added, which is as follows:

$$\begin{gathered} min(\widetilde{b}^k)^T \widetilde{\varphi}^{(k)} \widetilde{b}^k \\ s.t. \widetilde{b}^k \in 0, 1^{|U^S+1|}, b^k_{|U^S+1|} = 1, \sum_{i=1}^{|U^S+1|} \widetilde{b}_i^{\,k} = \lceil |U^S + 1|/2 \rceil \end{gathered} \tag{12}$$

Finally, by performing Cholesky decomposition, Eq. 12 can be transformed to an clustering problem, which can be formulated as:

$$\begin{aligned} &\min \sum_{u \in U'} ||u - \frac{1}{\lceil M/2 \rceil} \sum_{v \in U'} ||^2 \\ &s.t. U' \subseteq U, |U'| = \lceil M/2 \rceil, u_{\lceil M/2 \rceil} \in U' \end{aligned} \tag{13}$$

where U is the dataset, and a subset U with size $\lceil M/2 \rceil$ is desired where the square distances within is minimized. The 2-proximation algorithm is applied to solve Eq. 13.

Table 2: The overview of the integration of MLSH.

Input:	Q hash tables $\{F^{\{1\}}, F^{\{2\}}, ..., F^{\{Q\}}\}$, a query image I_q, the Hamming distance threshold H_{thres}.
	1. Generate hash codes for I_q, $B_q = \{B^{\{1\}}, B^{\{2\}}, ..., B^{\{Q\}}\}$ 2. According to Q hash tables, images within Hamming distance H_{thres} are returned to form Q returned image sets $S = \{S^{\{1\}}, S^{\{2\}}, ..., S^{\{Q\}}\}$. 3. The final returned image set is formed yielding $S_q = S^{\{1\}} \cap S^{\{2\}} \cap, ..., \cap S^{\{Q\}}$.
Output:	The final returned image set.

3.3. Integration of Multiple Hash Tables

As is shown in Table 2, images belonging to the intersection of the images returned by Q hash tables are finally returned. Given a query image, a subset of images yielding the least Hamming distance with the query are returned. Different hash tables contain different image information, such as structure information and semantic information. Images of different subset intersections are to be returned to guarantee the preservation of both semantic information and structure. Because the fusion of different features to train a single hash table leads to the similarity preservation of semantic and structure weaken each other, the proposed method can effectively eliminate the shortcomings of feature fusion.

Moreover, given a query, images within Hamming distance h are returned, which forms a Hamming ball with radius h. Normally, the larger Hamming ball is formed, when more similar images are desired. It unavoidably returns more dissimilar images, dragging down the precision. The proposed method only returns the intersection of multiple Hamming ball, which can return more similar images and meanwhile better remove part of dissimilar images. Precision and recall can be balanced in this way.

3.4. Complexity Analysis

The time complexity of the feature extraction CNN is $O(\sum_{l=1}^{D} M_l^2 K_l^2 C_{l-1} C_l)$, where M_l, D, K_l, and C_l is the size of feature map in the l^{th} layer, the number of layers, the kernel size of the l^{th} layer, and the channel of the l^{th} layer, respectively. M is $((X - K + 2Padding)/Stride + 1)$. The space complexity of CNN is $O(\sum_{l=1}^{D} K_l^2 C_{l-1} C_l)$. The time complexity of COSDISH is $O(T_{sto} \times T_{alt} \times (Nq^2 + q^4))$, where T_{sto}, T_{alt}, N, and qis the number of iterations, the alternative optimization iterations, the number of images, and the code length, respectively. The space complexity of COSDISH is $O(Nq + q^2)$. After the feature extraction, multiple hash tables can be trained simultaneously. The overall space complexity of hash table training is $O(Q \times (Nq + q^2))$, where Q is the number of hash tables. The complexity of hash code generating is $O(\sum_{t=1}^{Q} N \times d_t \times q)$, where d_t is the feature dimension for the t^{th} hash table.

4. Experiments

In our work, the experiments are conducted on three widely-used image databases, comparing the proposed method to several state-of-the-art hashing methods.

4.1. Databases and Settings

The three databases are CIFAR-10, MNIST and NUSWIDE. There are $60,000$ 32×32 color images from 10 categories in cifar-10, where $6,000$ images for each category with a single label. Following [26, 27], $50,000$images are

Table 3: Mean Average Precision (MAP) of hashing with different number of hash bits on CIFAR-10.

Methods	16-bit	24-bit	32-bit	48-bit	64-bit
MLSH	4×16-bit	4×24-bit	4×32-bit	4×48-bit	4×64-bit
	74.75%	**75.39**%	**75.57**%	**75.99**%	**76.00**%
MLSH	4×4-bit	4×6-bit	4×8-bit	4×12-bit	4×16-bit
	66.69%	69.71%	71.88%	73.76%	74.75%
CODISH [25]	57.12%	60.41%	61.57%	63.25%	63.77%
DDH [30]	60.35%	61.65%	63.21%	63.66%	62.92%
DSH [29]	63.66%	65.12%	66.07%	67.55%	-
DRSCH [31]	61.46%	62.19%	62.87%	63.05%	63.26%
DSRH [28]	60.84%	61.08%	61.74%	61.77%	62.91%
NINH [27]	55.40%	56.60%	58.80%	58.10%	-
CNNH [26]	44.10%	51.10%	50.90%	52.20%	-
BRE-CNN	19.80%	20.57%	20.59%	21.64%	21.96%
KSH-CNN	40.08%	42.98%	44.39%	45.77%	46.56%
MLH-CNN	25.04%	28.86%	31.29%	31.88%	31.83%
BRE [24]	12.19%	15.63%	16.10%	17.19%	17.56%
KSH [23]	32.15%	35.17%	36.51%	38.26%	39.50%
MLH [22]	13.33%	15.78%	16.29%	18.03%	18.84%
ITQ [17]	11.45%	11.63%	11.53%	10.97%	11.24%
SH [18]	19.22%	19.28%	20.09%	20.79%	21.46%
LSH [1]	12.36%	11.74%	12.30%	13.57%	12.42%

chosen as the training set and $10,000$ images as the query set. The raw pixel-based images are used as the input for the proposed method. The traditional hashing methods employs the 512-dimension GIST feature as the input.

MNIST contains $70,000$ 28×28 gray-scale images from 10 categories (i.e. digits from 0 to 9). It is a handwriting digits image database. Following [26, 27], $60,000$ images are chosen to build the training set and $10,000$ images for the query set. For the proposed method, the raw pixel-based images are used as the input. For the traditional hashing methods, the SIFT feature representation is used as the input.

NUSWIDE contains $269,648$ resized 64×64 color images from 81 categories. Each image belongs to one or multiple categories. Following [26, 27], 21 the most

Table 4: Mean Average Precision (MAP) of hashing with different number of hash bits on MNIST.

Methods	16-bit	24-bit	32-bit	48-bit	64-bit
MLSH	4×16-bit	4×24-bit	4×32-bit	4×48-bit	4×64-bit
	99.52%	**99.53**%	**99.54**%	**99.55**%	**99.63**%
MLSH	4×4-bit	4×6-bit	4×8-bit	4×12-bit	4×16-bit
	99.20%	99.34%	99.43%	99.53%	99.52%
CODISH [25]	86.17%	87.27%	86.77%	87.99%	87.91%
DDH [30]	98.48%	98.61%	98.86%	98.67%	98.89%
DSH [29]	98.92%	99.02%	99.11%	99.20%	99.20%
DRSCH [31]	96.92%	97.37%	97.88%	97.91%	98.09%
DSRH [28]	96.48%	97.79%	97.21%	97.53%	97.75%
NINH [27]	93.90%	94.90%	95.80%	95.90%	-
CNNH [26]	93.40%	95.50%	96.40%	96.50%	-
BRE-CNN	61.00%	64.05%	64.11%	66.33%	67.02%
KSH-CNN	83.89%	86.67%	88.51%	89.41%	89.67%
MLH-CNN	71.03%	76.18%	78.06%	80.66%	80.87%
BRE [24]	41.96%	57.19%	56.52%	64.74%	66.55%
KSH [23]	82.85%	86.03%	87.37%	88.48%	88.82%
MLH [22]	45.77%	62.16%	63.07%	65.23%	66.70%
ITQ [17]	34.44%	38.99%	40.62%	43.04%	41.76%
SH [18]	13.40%	14.81%	15.28%	16.29%	17.11%
LSH [1]	22.65%	21.39%	35.56%	27.85%	37.78%

commonly-used categories are used, including over $5,000$ images per category. Some images urls are invalid from the Internet, so there are $157,465$ images in total. 500 images and 100 images per category are selected to build the training set and the query set, respectively. The training set and the query set consist of $10,500$ images and $2,100$ images in total. The pixel-based images are used as the input of the proposed hashing network for the proposed method. For the traditional hashing methods, the GIST feature serves as the input. For NUSWIDE database, only the top $5,000$ returned images are used to compute the MAP score.

For fair comparison, the $4,096$-dimension deep feature extracted from the AlexNet [46] also serves as the alternative feature representation for the tradi-

Table 5: Mean Average Precision (MAP) of hashing with different number of hash bits on NUSWIDE.

Methods	16-bit	24-bit	32-bit	48-bit	64-bit
MLSH	4 × 16-bit	4 × 24-bit	4 × 32-bit	4 × 48-bit	4 × 64-bit
	70.88%	**71.53**%	**72.61**%	**72.93**%	**73.12**%
MLSH	4 × 4-bit	4 × 6-bit	4 × 8-bit	4 × 12-bit	4 × 16-bit
	64.26%	64.56%	67.02%	68.72%	70.88%
CODISH [25]	60.04%	61.07%	62.99%	63.65%	64.29%
DDH [30]	54.21%	55.20%	56.86%	57.49%	56.82%
DSH [29]	54.97%	55.13%	55.82%	56.21%	-
DRSCH [31]	61.81%	62.24%	62.27%	62.79%	64.14%
DSRH [28]	60.92%	61.78%	62.13%	63.09%	64.02%
NINH [27]	68.10%	69.70%	71.30%	71.50%	-
CNNH [26]	62.30%	61.80%	62.50%	60.80%	-
BRE-CNN	53.80%	55.79%	56.58%	57.58%	59.13%
KSH-CNN	60.74%	61.89%	62.46%	62.57%	63.11%
MLH-CNN	52.51%	55.91%	56.83%	58.07%	59.79%
BRE [24]	48.64%	51.45%	51.83%	52.75%	54.66%
KSH [23]	54.56%	55.63%	56.22%	56.68%	58.35%
MLH [22]	48.71%	50.69%	51.11%	52.38%	54.03%
ITQ [17]	45.23%	46.14%	46.71%	47.07%	47.29%
SH [18]	43.33%	43.26%	43.81%	43.06%	45.18%
LSH [1]	40.18%	41.88%	42.26%	42.04%	45.48%

tional hashing methods.

The experiment firstly compares the proposed method to several traditional hashing methods, including LSH [1], SH [18], ITQ [17], MLH [22], KSH [23], BRE [24], and CODISH [25]. CODISH is also the basic supervised hashing method in our proposed method. Moreover, for fair comparison, CNN features are applied to three traditional hashing methods, i.e. BRE-CNN, KSH-CNN, and MLH-CNN, respectively. Also, the proposed method is compared to several deep hashing methods, including CNNH [26], NINH [27], DSRG [28], DRSCH [31], DSH cite20, and DDH [30]. The MAP score is used as the measurement to evaluate the retrieval performance based on the semantic ground-truth.

4.2. Comparison with State-of-the-art Hashing

The comparison MAP scores on CIFAR-10, MNIST and NUSWIDE are presented in Tables 3 - 5. The cell marked with '-' indicates the corresponding results are not provided by the original paper. '4×4'-bit denotes the integration of four 4-bit hash tables. The cell with the bold font is the largest value and with the Italic font is the second largest value. In addition, the proposed method is tested with several settings and the results are shown in two rows in each table. The results of the proposed method when the hash code length of every individual hash table is the same as the comparison methods are presented in the first row. To further demonstrate the storage efficiency of the proposed method, the second row presents the result of the proposed method when the total hash code length is the same as that of the comparison methods.

Table 6: Mean Average Precision (MAP) of hashing of different scales with different number of hash bits on CIFAR-10.

MLSH	4-bit	8-bit	16-bit	24-bit	32-bit	48-bit	64-bit
fc_1	51.45%	63.55%	68.48%	71.44%	71.62%	72.10%	72.66%
$conv_3$	47.92%	61.57%	66.68%	69.15%	69.86%	70.63%	71.87%
$conv_2$	38.07%	56.95%	62.06%	64.74%	65.94%	66.27%	67.43%
$conv_1$	41.49%	43.53%	57.83%	60.93%	62.52%	63.59%	64.17%
	4×4-bit	4×8-bit	4×16-bit	4×24-bit	4×32-bit	4×48-bit	4×64-bit
4 tables	**66.69**%	**71.88**%	**74.75**%	**75.39**%	**75.57**%	**75.99**%	**76.00**%
	-	-	4×4-bit	4×6-bit	4×8-bit	4×12-bit	4×16-bit
4 tables	-	-	66.69%	69.71%	71.88%	73.76%	74.75%

Clearly, the integration of four hash tables of different levels is proven to acquire better retrieval performance than other traditional hashing methods, even the deep hashing methods. For example, on CIFAR-10, the proposed method can achieve a MAP of 74.75% when the code length for each table is 16 bits, while the second largest MAP is 63.66% On MNIST, compared to the second best method DSH, the MAP score of the proposed method exceeds by 0.6% using 16 bits per table. On NUSWIDE, compared to the second best method NINH, the MAP of the proposed method exceeds by 2.78% using 16

Table 7: Mean Average Precision (MAP) of hashing of different scales with different number of hash bits on MNIST.

MLSH	4-bit	8-bit	16-bit	24-bit	32-bit	48-bit	64-bit
fc_1	98.25%	98.96%	99.26%	99.31%	99.34%	99.44%	99.44%
$conv_3$	87.31%	98.16%	98.71%	98.69%	98.98%	98.93%	99.15%
$conv_2$	87.08%	97.87%	98.33%	98.61%	98.69%	98.94%	98.85%
$conv_1$	84.99%	96.95%	97.67%	97.80%	97.98%	98.26%	98.32%
	$4\times$4-bit	$4\times$8-bit	$4\times$16-bit	$4\times$24-bit	$4\times$32-bit	$4\times$48-bit	$4\times$64-bit
4 tables	**99.20**%	**99.43**%	**99.52**%	**99.53**%	**99.54**%	**99.55**%	**99.63**%
	-	-	$4\times$4-bit	$4\times$6-bit	$4\times$8-bit	$4\times$12-bit	$4\times$16-bit
4 tables	-	-	99.20%	99.34%	99.43%	99.53%	99.52%

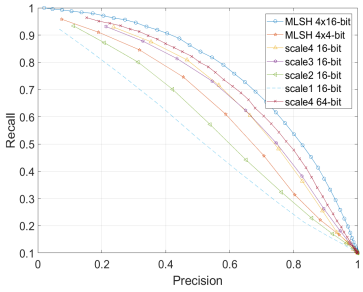
Table 8: Mean Average Precision (MAP) of hashing of different scales with different number of hash bits on NUSWIDE.

MLSH	4-bit	8-bit	16-bit	24-bit	32-bit	48-bit	64-bit
fc_1	54.58%	53.97%	49.11%	61.87%	60.50%	62.92%	71.51%
$conv_3$	53.16%	59.00%	57.96%	56.16%	58.71%	59.52%	65.78%
$conv_2$	50.56%	50.99%	54.95%	57.75%	60.96%	53.73%	63.85%
$conv_1$	50.08%	51.97%	53.07%	49.87%	60.16%	59.14%	58.23%
	$4\times$4-bit	$4\times$8-bit	$4\times$16-bit	$4\times$24-bit	$4\times$32-bit	$4\times$48-bit	$4\times$64-bit
4 tables	**64.26**%	**67.02**%	**68.72**%	**70.88**%	**71.53**%	**72.61**%	**72.93**%
	-	-	$4\times$4-bit	$4\times$6-bit	$4\times$8-bit	$4\times$12-bit	$4\times$16-bit
4 tables	-	-	64.26%	64.56%	67.02%	68.72%	70.88%

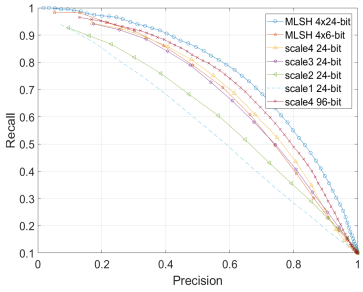
bits per hash table. Moreover, with the same total storage cost, the proposed method outperforms other comparison methods, which proves to have storage efficiency. For example, the integration of four-level 4-bit hash table achieves a MAP of 66.69%, which exceeds by 3.03% compared to the best competitor DSH on CIFAR-10. In addition, compared to the basic hashing method CODISH with traditional hand-crafted feature, the proposed method with multi-level deep feature achieves better retrieval performance.

4.3. Validation of the Multi-Level Mechanism

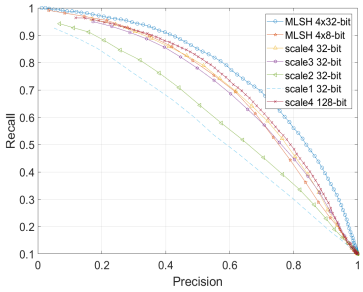
With the same basic hashing method, using the integration of multi-level features achieves better performance than using a single-level representation.



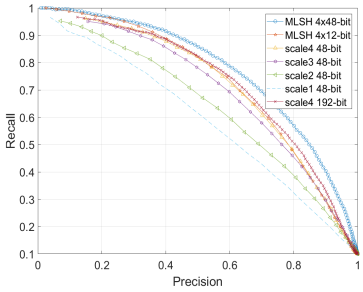
(a) 16 bits



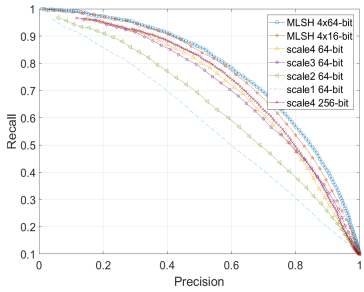
(b) 24 bits



(c) 32 bits



(d) 48 bits



(e) 64 bits

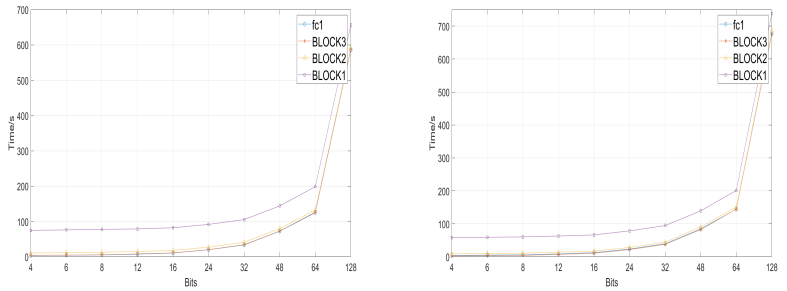
Figure 3: Recall-Precision curve of hashing of different scales with different number of hash bits on CIFAR-10.

Figure 4: Top 10 retrieval results with 32-bit hash table using feature of the corresponding scale, and the final retrieval results of MLSH on NUSWIDE. (Image with green border is the positive instance, while image with red border is negative instance.)

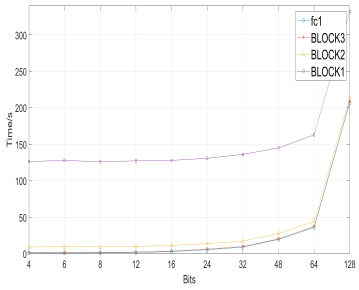
Tables 6 - 8 present the retrieval performance of using single-level features and multi-level features on CIFAR-10 MNIST and NUSWIDE, respectively. The row beginning with the cell marked with '$fc1$' presents the MAP scores of using the feature extracted from the first fully connected layer in the network with different hash bits. Similarly, '$conv3$' denotes the third convolutional layer in the network. In each table, the MAP scores of using the features from the corresponding level (i.e. $fc1$, $conv3$, $conv2$, or $conv1$) are shown as a comparison, and the MAP scores of the proposed method, which is the integration of the hash tables using the above four-level features are shown in the bottom. '4×4-bit' denotes the integration of four hash tables, each of which is with 4 bits. From the table, it is obvious that among the single-level methods, using the feature from the penultimate layer (i.e. $fc1$) achieves the best performance. For example, on CIFAR-10 with 4 bits, using feature from '$fc1$' achieves the largest MAP score of 51.45%, while that from '$conv3$' only achieves 47.92%. Moreover, to reveal the efficiency of the proposed method, the MAP scores of the integration of the hash tables using the above four-level features is calculated. For example, in the case of using 4 bits for an individual table, the integration

of four hash tables of corresponding four levels achieves the MAP of 66.69%, which is 25.2% higher than that of simply using single-level features. In order to maintain the same total storage cost with the single-level hashing methods, the code length of each table is shortened. For example, the MAP of using the integration of four 4-bit hash tables exceeds that of using a single 16-bit hash table by 8.86%. Fig. 3 presents the retrieval performance of using single-level features and multi-level features on CIFAR-10. Take Fig. 3(a) as an example, 'MLSH $4 \times$ 16-bit' indicates that the proposed method integrates four 16-bit hash tables corresponding to different levels. 'level4 16-bit' denotes that the baseline hashing method trains only a single 16-bit table using the feature at the 4th level. Compared to the single-level methods, MLSH is proved to achieve higher recall and precision, with the same code length in each individual hash table. Furthermore, with the code length increasing, the performance improvement of the single-level hashing will slow down. With the same total storage cost, MLSH ($4 \times$16-bit) preforms better than the baseline method with a single 64-bit hash table. From Fig. 3(c) - 3(e), MLSH will become more and more advantageous as the code length becomes longer.

The proposed multi-level hashing method is more prominent in its superiority on multi-label datasets. As is shown in Table 7, features from the higher level cannot guarantee a better retrieval performance. We visualized a retrieval process on the NUSWIDE dataset, which is illustrated in Fig. 4. When a query with multiple labels (i.e. person, vehicle and window) is given, top 10 images are returned using a 32-bit hash table using the features of the corresponding level. MLSH returns the intersection of multi-level hash table retrieval results. All of the top 10 retrieval results using $fc1$ features share no less than one label with the query image. However, excessive attention to semantic information may neglect the detailed features. Therefore, combined with the use of low-level features to extract detailed information, the retrieved images can be more similar in both structure and semantics with the query image.



(a) CIFAR-10 (b) MNIST



(c)

Figure 5: Training time of the proposed method on (*a*)CIFAR-10, (*b*)MNIST and (*c*)NUSWIDE.

4.4. Speed Analysis

Fig. 5 shows the time cost of training hash tables in MLSH, which varies with different code lengths and feature dimensions. Training time of hash table for different level of feature increases linearly corresponding to the feature dimension. Moreover, with the hash code length increases, the training time increase exponentially.

5. Conclusion

A novel image hashing method, MLSH, is proposed in this paper, which applies a multiple-hash-table mechanism to integrate multiple levels of features to preserve both semantic and structural information. One hash table is trained

with a level of deep features extracted from a network layer, which contains a different level of information. Experimental results demonstrate a better retrieval performance of the proposed method comparing to other state-of-the-art hashing methods.

The proposed method generates hash code with both semantic similarity preserving and structural similarity preserving. Besides, it can better balance recall and precision. However, it does not consider the bit-wise difference and redundancy. Moreover, in MLSH, multiple hash tables based on various levels of features are integrated equally in a bagging manner, where features of different levels may contain redundant information. In future work, multiple hash tables can be trained in a complementary boosting manner so that the following level hash table can correct the errors generated from the previous level hash tables.

Moreover, the proposed method individually extracts multi-level features and trains corresponding hash tables, in which hash code learning cannot provide feedback to the feature extraction. In future work, the multi-level hashing scheme may be extended to an end-to-end system of deep hashing method so that hash code learning is capable of giving feedback to the multi-level feature extraction process.

ACKNOWLEDGMENT

This work was supported in part by the National Natural Science Foundation of China under Grant 61876066, Grant 61572201, Grant 61772344, and Grant 61672443, in part by the Guangzhou Science and Technology Plan Project under Grant 201804010245, and EU Horizon 2020 Programme (700381, ASGARD), and in part by the Hong Kong RGC General Research Funds under Grant 9042489 (CityU 11206317), Grant 9042816 (CityU 11209819) and Grant 9042322 (CityU 11200116).

References

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

[2] M. Raginsky, S. Lazebnik, Locality-sensitive binary codes from shift-invariant kernels, in: Proc. Conf. and Workshop on Neural Information Processing Systems, 2009, pp. 1509–1517.

[3] B. Kulis, K. Grauman, Kernelized locality-sensitive hashing for scalable image search, in: Proc. International Conference on Computer Vision, 2009, pp. 1707–1720.

[4] J. Wang, T. Zhang, j. song, N. Sebe, H. T. Shen, A survey on learning to hash, IEEE Transactions on Pattern Analysis and Machine Intelligence 40 (4) (2018) 769–790.

[5] G. Dniz, J. Bueno, Salido, F. D. la Torre, Face recognition using histograms of oriented gradients, Pattern Recognition.Letter 32 (12) (2011) 1598–1603.

[6] D. Huang, C. Shan, M. Ardabilian, Y. Wang, L. Chen, Local binary patterns and its application to facial image analysis: A survey, IEEE Trans. on Systems, Man, and Cybernetics 41 (6) (2011) 765–781.

[7] V. Purandare, K. T. Talele, Efficient heterogeneous face recognition using scale invariant feature transform, in: Proc. International Conference on Circuits, Systems, Communication and Information Technology Applications, 2014, pp. 305–310.

[8] A. Oliva, A. Torralba, Modeling the shape of the scene: A holistic representation of the spatial envelope, International Journal of Computer Vision 42 (2001) 145–175.

[9] F. Sabahi, M. O. Ahmad, M. N. S. Swamy, Content-based image retrieval using perceptual image hashing and hopfield neural network, in: Proc. 2018

IEEE 61st International Midwest Symposium on Circuits and Systems, 2018, pp. 352–355.

[10] A. Babenko, A. Slesarev, A. Chigorin, V. S. Lempitsky, Neural codes for image retrieval, in: Proc. European Conference on Computer Vision, 2014, pp. 584–599.

[11] Y. Gong, L. Wang, R. Guo, S. Lazebnik, Multi-scale orderless pooling of deep convolutional activation features, in: Proc. European Conference on Computer Vision, 2014, pp. 392–407.

[12] S. Kim, S. Choi, Multi-view anchor graph hashing, IEEE Int. Conf. Acoust., Speech Signal Process. (2013) 3123–3127.

[13] D. Zhang, F. Wang, L. Si, Composite hashing with multiple information sources, in: Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., 2011, pp. 225–234.

[14] S. Kim, Y. Kang, S. Choi, Sequential spectral learning to hash with multiple representations, in: Proc. European Conference of Computer Vision, 2012, pp. 538–551.

[15] D. Wang, P. Cui, M. Ou, W. Zhu, Deep multimodal hashing with orthogonal regularization, in: Proc. International Conference on Artificial Intelligence, 2015, pp. 2291–2297.

[16] P. Li, J. Cheng, H. Lu, Hashing with dual complementary projection learning for fast image retrieval, Neurocomputing 120 (2013) 83–89.

[17] Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval, IEEE Trans. on Pattern Anal. Mach. Intell. 35 (12) (2013) 2916–2929.

[18] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: Proc. Neural Information Processing Systems, 2009, pp. 1753–1760.

[19] Y. Lv, W. W. Y. Ng, Z. Zeng, D. S. Yeung, P. P. K. Chan, Asymmetric cyclical hashing for large scale image retrieval, IEEE Transactions on Multimedia 17 (8) (2015) 1225–1235.

[20] J. Wang, S. Kumar, S.-F. Chang, Semi-supervised hashing for scalable image retrieval, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2010.

[21] C. Wu, J. Zhu, D. Cai, C. Chen, J. Bu, Semi-supervised nonlinear hashing using bootstrap sequential projection learning, IEEE Trans. Knowl. Data Eng. 25 (6) (2013) 1380–1393.

[22] M. Norouzi, D. J. Fleet, Minimal loss hashing for compact binary codes, in: Proc. International Conference on Machine Learning, 2011, pp. 353–360.

[23] W. Liu, J. Wang, R. Ji, Y. G. Jiang, S. F. Chang, Supervised hashing with kernels, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 2074–2081.

[24] B. Kulis, T. Darrell, Learning to hash with binary reconstructive embeddings, in: Proc. Neural Information Processing Systems, 2009, pp. 1042–1050.

[25] W.-C. Kang, W.-J. Li, Z.-H. Zhou, Column sampling based discrete supervised hashing, the Association for the Advance of Artificial Intelligence.

[26] R. Xia, Y. Pan, H. Lai, C. Liu, S. Yan, Supervised hashing for image retrieval via image representation learning, in: Proc. Conference on Artificial Intelligence, 2014, pp. 2156–2162.

[27] H. Lai, Y. Pan, Y. Liu, S. Yan, Simultaneous feature learning and hash coding with deep neural networks, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3270–3278.

[28] T. Yao, F. Long, T. Mei, Y. Rui, Deep semantic-preserving and ranking-based hashing for image retrieval, in: Proc. International Joint Conference on Artificial Intelligence, 2016, pp. 3931–3937.

[29] H. Liu, R. Wang, S. Shan, X. Chen, Deep supervised hashing for fast image retrieval, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2064–2072.

[30] J. Lin, Z. Li, J. Tang, Discriminative deep hashing for scalable face image retrieval, in: Proc. International Joint Conference on Artificial Intelligence, 2017, pp. 2266–2272.

[31] R. Zhang, L. Lin, R. Zhang, W. Zuo, L. Zhang, Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification, IEEE trans. on Image Processing 24 (12) (2015) 4766–4779.

[32] J. Zhang, Y. Peng, Query-adaptive image retrieval by deep-weighted hashing, IEEE Transactions on Multimedia 20 (9) (2018) 2400–2414.

[33] J. Li, W. W. Y. Ng, T. Xing, S. Kwong, H. Wang, Weighted multi-deep ranking supervised hashing for efficient image retrieval, IEEE Transactions on Multimedia.

[34] Z. Yu, W. Wang, Learning dalts for cross-modal retrieval, CAAI Transactions on Intelligence Technology 4 (1) (2019) 9–16.

[35] C. Yan, L. Li, B. L. C. Zhan and, Y. Zhang, Q. Dai, Cross-modality bridging and knowledge transferring for image understanding, IEEE Transactions on Multimedia 21 (10) (2019) 2675–2685.

[36] S. Bell, C. L. Zitnick, K. Bala, R. Girshick, Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2874–2883.

[37] W. Zhao, L. Yan, Y. Zhang, Geometric-constrained multi-view image matching method based on semi-global optimization, Geo-spatial Information Science 21 (2) (2018) 115–126.

[38] A. Liu, Y. Qiu, Y. Wong, Y. Su, M. Kankanhalli, A fine-grained spatial-temporal attention model for video captioning, IEEE Access 6 (2018) 68463–68471.

[39] J. H. et al., Mfc: A multi-scale fully convolutional approach for visual instance retrieval, in: Proc. IEEE International Conference on Multimedia & Expo Workshops, IEEE Computer Society, 2017, pp. 513–518.

[40] J. Y. Ng, F. Yang, L. S. Davis, Exploiting local features from deep networks for image retrieval, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015, pp. 53–61.

[41] B. Artem, L. Victor, Aggregating deep convolutional features for image retrieval, in: Proc. IEEE Conference on Computer Vision and Pattern Recognition, 2015.

[42] X. Lu, Y. Chen, X. Li, Hierarchical recurrent neural hashing for image retrieval with hierarchical convolutional features, IEEE Transactions on Image Processing 27 (1) (2018) 106–120.

[43] J. Wang, S. Kumar, S.-F. Chang, Exploiting the complementary strengths of multi-layer cnn features for image retrieval, Neurocomputing (2017) 235–241.

[44] L. M. Abualiga, A. T. Khader, Unsupervised text feature selection technique based on hybrid particle swarm optimization algorithm with genetic operators for the text clustering, supercomputing 73 (11) (2017) 4773–4795.

[45] L. Abualigah, Feature selection and enhanced krill herd algorithm for text document clustering.

[46] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proc. Neural Information Processing Systems, 2012, pp. 1097–1105.