Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 33(7): 4151–4164.
DOI: 10.3934/era.2025186
Received: 14 May 2025
Revised: 13 June 2025
Accepted: 08 July 2025
Published: 09 July 2025

Research article

Large-scale content-based image retrieval system with metric learning and discrete binary encoding

Yahui Liu*, Jianbo Dai, Liang Hu and Guangxu Zhao

China Coal Technology and Engineering Group Chongqing Research Institute, Chongqing 400039, China

* **Correspondence:** Email: liuyahuiaa@163com.

Abstract: Hashing methods have been widely used in large-scale content-based image retrieval systems, which have extensive and practical applications in search engines and social networks. However, existing deep supervised hashing methods focus on enforcing the supervision information to learn the discrete hash codes, ignoring the latent clustering in the Hamming space. In addition, the discrete constraints in the above optimization problem are usually hard to solve. To address these issues, we propose a new deep supervised hashing method for image retrieval tasks. Specifically, we utilized a metric learning strategy to minimize the intra-class Hamming distance and maximize the inter-class Hamming distance as much as possible. Then, a novel label fitting approach was developed to generate discrete proxies and hash codes simultaneously. Finally, an effective alternating optimization method was designed to solve the discrete optimization problem. Experimental results on various datasets demonstrate that the proposed method can achieve promising retrieval performance.

Keywords: image retrieval; metric learning; deep learning; discrete binary encoding

1. Introduction

The explosive growth of data in search engines and social networks poses a great challenge for retrieving similar images from a large dataset. Recently, the approximate nearest neighbors [1–5] (ANN) search technique has received increased attention due to the fast retrieval speed and low memory consumption. Learning to hash methods [6], as the most representative ANN search technology, has received great attention in large-scale content-based image retrieval systems. The goal of learning to hash is to project the original high-dimensional features into a compact binary code while preserving data similarities. As is known, Hamming ranking is achieved through the XOR and POPCNT operations on modern CPUs or GPUs, which can reach a constant or sub-linear search speed. Therefore, efficient storage and searching make hashing technology popular for large-scale content-based image retrieval systems.

Recently, deep supervised hashing methods have integrated supervised information, including category labels, semantic similarities, and ranking, into the feature learning and hash encoding, which became very popular in large-scale content-based image retrieval systems, due to the promising retrieval. Typical deep supervised hashing methods include convolutional neural network hashing (CNNH) [7], network in network hashing (NINH) [8], deep semantic ranking hashing (DSRH) [9], deep similarity comparison hashing (DSCH) [10], deep pairwise supervised hashing (DPSH) [11], deep supervised hashing (DSH) [12], deep hashing network (DHN) [13], deep quantization network (DQN) [14], and HashNet [15]. CNNH [7] utilizes a two-step method to learn hash functions, i.e., learning approximate hash codes by using a relaxation strategy in the first step and training the network with the approximate hash codes and the label information in the second step. However, in this two-step method, the learned approximate hash codes can be utilized to guide the feature learning procedure, but not the reverse. Therefore, most existing deep supervised hashing methods [8–15] perform feature learning and hash coding simultaneously. Under this architecture, the learned feature representation can provide a feedback for learning hash codes and vice versa. Recent works [8, 11–13] showed that this architecture performs better than the two-step method [7].

The main challenge in existing deep supervised hashing methods is to solve the mix-integer optimization problems formed by imposing discrete constraints on the binary codes. Generally, the mix-integer optimization problems are NP-hard. Many deep supervised hashing methods [8–10] choose to solve a relaxed problem by removing the discrete constraints directly. However, they are less effective for hashing retrieval tasks due to the large quantization errors. In order to reduce the quantization error, some deep supervised hashing methods [12, 13, 15] propose a set of methods to approximate the discrete constraints. For example, DSH [12] utilizes L1-norm to replace the binary constraints. DHN [13] replaces the L1-norm in [12] with a smooth surrogate. HashNet [15] adopts a sequence of continuous relaxations to approach the discrete constraints. These methods can only obtain binary-like codes in the training procedure. Recent works [11, 14] focused on learning the discrete hash codes without relaxations. For example, DPSH [11] resorts to minimizing the quantization error between the outputs of the deep neural network and the desired binary codes. DQN [14] introduces a product quantization loss over the outputs of the deep neural network to learn discrete binary codes of training data points. From the point of view of supervision information usage, it directly guides the deep feature learning procedure but not the hash coding procedure. In other words, the supervision information is only included in the term for learning better image representation but not in the term for learning better hash codes. Recently, CMH [16], MDSH [17], and CSQ [18] have been used to attempt to learn semantic hash centers and achieve promising image retrieval performance with more complex network structure.

In this paper, we propose a new deep supervised hashing method for image retrieval that learns an end-to-end architecture using the supervision information to learn the deep feature, discrete proxies, and hash codes simultaneously. Existing hashing methods focus on enforcing the supervision information to learn the discrete hash codes only, while ignoring the latent clustering in the Hamming space. We apply the supervision information to generate discrete proxies and binary codes simultaneously. Specifically, we utilize a metric learning strategy to minimize the intra-class Hamming distance and maximize the inter-class Hamming distance as much as possible. Second, a novel label fitting approach is developed to generate discrete proxies and hash codes simultaneously. Finally, an effective alternating optimization method is designed to solve the discrete optimization problem. The proposed alternating optimization method under a stochastic gradient descent framework can obtain discrete proxies and binary codes,

which can circumvent the large binary quadratic programming problem and speed up the training procedure. Experimental results on various datasets demonstrate that the proposed method can achieve promising retrieval performance for large-scale content-based image retrieval systems.

2. Notation and problem definition

2.1. Notation

Throughout this paper, we use boldface lowercase letters like $\mathbf{a}$ and boldface uppercase letters like $\mathbf{A}$ to denote vectors and matrices, respectively. The i-th element in vector $\mathbf{a}$ is denoted as a_i. Let A_{ij} denote the (i, j)-th observation in the i-th row and j-th column of matrix $\mathbf{A}$. $\mathbf{a}_j$ denotes the j-th column of matrix $\mathbf{A}$ and $a_{ij} = A_{ij}$. Let $\|\cdot\|_F$ denote the Frobenius norm of a vector or matrix. The symbol $\odot$ is used to denote the element-wise product (i.e., Hadamard product). The sign$(\cdot)$ denotes the element-wise sign function.

2.2. Problem definition

Assume we have a training set $\mathbf{X} = \{\mathbf{x}_i, y_i\}_{i=1}^n$ consisting of n data points, where $\mathbf{x}_i$ denotes the i-th observation data, and $y_i \in \mathbb{R}^l$ denotes its label. Moreover, we define a supervision matrix $\mathbf{S} \in \{-1, 1\}^{n\times n}$ based on the semantic similarity. With this pairwise supervised information $\mathbf{S}$, supervised hashing aims to learn a hash function to transform the data from the original high-dimensional space into a compact binary space while preserving semantic similarities in the binary space. The corresponding hashing function is denoted as $h(\mathbf{x}) \in \{-1, +1\}^q$, where q is the desired code length. Let us stack the desired hash codes column by column and form the resulting code matrix $\mathbf{B} = [\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n] \in \{-1, 1\}^{q\times n}$, where the hash code $\mathbf{b}_j = h(\mathbf{x}_j)$ of $\mathbf{x}_j$ corresponds to the elements in the j-th column of $\mathbf{B}$. Therefore, we can calculate the Hamming distance between $\mathbf{b}_i$ and $\mathbf{b}_j$ as follows:

$$d_{\mathcal{H}}(\mathbf{b}_i, \mathbf{b}_j) = \frac{1}{4}\|\mathbf{b}_i - \mathbf{b}_j\|_F^2 = \frac{q}{2} - \frac{1}{2}\mathbf{b}_i^T\mathbf{b}_j. \tag{2.1}$$

In order to preserve the semantic similarity in $\mathbf{S}$, we expect the Hamming distance between semantically similar pairs to be small, and the Hamming distance to be large for semantically dissimilar pairs. If we use all the supervised information in $\mathbf{S}$ to train directly, the training time and space complexity will be greater than $O(n^2)$. In real applications, a subset Φ of $\mathcal{N} = \{1, 2, \cdots, n\}$ is randomly sampled in each iteration [21]. Then, the index of the whole training set can be divided into two subsets, i.e., Φ and $\Delta = \mathcal{N} - \Phi$. Therefore, in each iteration, we can formulate the supervised hashing learning problem as the following optimization problem:

$$\min_h \mathcal{L}(h) = \sum_{i,j\in\Phi} \mathcal{L}_1(h(\mathbf{x}_i), h(\mathbf{x}_j); S_{ij}) + \alpha\frac{|\Phi|}{|\Delta|}\mathcal{L}_2(\mathbf{P}, \mathbf{B}^{\Delta}) + \beta\sum_{i\in\Phi}\mathcal{L}_3(h(\mathbf{x}_i), \mathbf{b}_i) \tag{2.2}$$

where $h_m(\mathbf{x}_i) = b_{mi}$. α and β are the balance hyper-parameters. $\frac{|\Phi|}{|\Delta|}$ is added to balance the size of these two terms. The first term, $\mathcal{L}_1$, is used to ensure that the intra-class Hamming distance is reduced and the inter-class Hamming distance is increased so that the discriminative power of learned binary codes can be enhanced. The second term, $\mathcal{L}_2$, is used to ensure that the intra-class similarity between each proxy and binary codes is maximized and the inter-class similarity between each proxy and binary codes is

minimized, so that each hash codes can carry sufficient supervised information for learning discriminative hash functions. The third term, $\mathcal{L}_3$, is a regularization term to reduce the quantization errors.

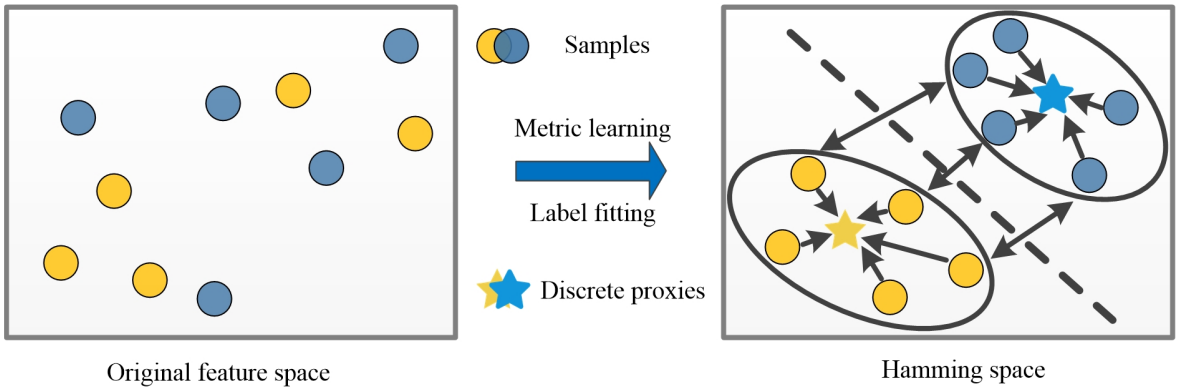


Figure 1. An illustration of the proposed method diagram.

3. Discrete binary encoding method with metric learning

In this section, we will introduce the proposed method in detail, which consists of two parts, i.e., model formulation and learning algorithm. Figure 1 gives an illustration of the proposed method diagram.

3.1. Model formulation

Due to the end-to-end framework, our method can simultaneously perform feature learning and hash coding with deep neural networks. A novel supervised loss is designed to learn optimal binary codes, which can preserve semantic similarities and learn latent discrete clustering (i.e., discrete proxies).

3.1.1. Configuration of the network

The proposed model contains a convolutional neural network (CNN) from [20] as a component for learning a deep feature representation. There are eight layers in our model, where the first seven are the same as those of CNN-F in [20]. Different from CNN-F, the eight layer of our model is a q-dimensional fully-connected layer. The detailed network configuration of our model is listed in Table 1. More specifically, the feature learning part consists of 5 convolutional layers (conv 1–5) and 3 fully-connected layers (fc 6–8). In Table 1, "filter" is denoted as "num × size × size", where "num" specifies the number of convolution filters and "size" indicates receptive field size. "stride" indicates the interval with which we slide the filter. "pad" specifies the number of pixels added around the border of the current layer input. "LRN" denotes whether local response normalization (LRN) is applied. "pool" specifies the downsampling factor. "4096" and "number of hash bits q" denote the number of nodes in that layer. The rectified linear unit (RELU) is utilized as the activation function for the first seven layers. For the last layer, we adopt the $\tanh(z)$ function as the activation function. The resulting binary codes are generated by the output of the deep feature learning part. Let $F(\mathbf{x};\Theta)$ denote the output of the deep feature learning part, where Θ is a set of network parameters. Therefore, the hash function can be defined as $h(\mathbf{x}) = \text{sign}(F(\mathbf{x};\Theta))$.

Table 1. Configuration of the CNN-F network.

All layers	Network configuration
conv1	filter $64 \times 11 \times 11$, stride 4×4, pad 0, LRN, pool 2×2
conv2	filter $256 \times 5 \times 5$, stride 1×1, pad 2, LRN, pool 2×2
conv3	filter $256 \times 3 \times 3$, stride 1×1, pad 1
conv4	filter $256 \times 3 \times 3$, stride 1×1, pad 1
conv5	filter $256 \times 3 \times 3$, stride 1×1, pad 1, pool 2×2
full6	the embedding dimension 4096
full7	the embedding dimension 4096
full8	the code-length q
activation function	$tanh(\cdot)$

3.1.2. Object function

According to the previous sampling scheme, the whole training set $\mathbf{X}$ can be split into two subsets $\mathbf{X}^{\Phi} = \{\mathbf{x}_i \mid i \in \Phi\}$ and $\mathbf{X}^{\Delta} = \{\mathbf{x}_i \mid i \in \Delta\}$. Let $\widetilde{\mathbf{S}} \in \{-1, 1\}^{|\Phi| \times n}$ denote the sampled sub-matrix of $\mathbf{S}$ with rows indexed by Φ, where $|\Phi|$ denotes the sampled row numbers. $\widetilde{\mathbf{S}}^{\Phi} \in \{-1, 1\}^{|\Phi| \times |\Phi|}$ denotes the sub-matrix of $\widetilde{\mathbf{S}}$ formed of columns indexed by Φ. We can use a similar way to define $\widetilde{\mathbf{S}}^{\Delta} \in \{-1, 1\}^{|\Phi| \times |\Delta|}$, $\mathbf{B}^{\Phi} \in \{-1, 1\}^{q \times |\Phi|}$, and $\mathbf{B}^{\Delta} \in \{-1, 1\}^{q \times |\Delta|}$.

Inspired by [22], we introduce a metric learning method for enhancing the discriminative ability of $\mathbf{B}^{\Phi}$. More specifically, we expect that the intra-class Hamming distance is as small as possible and the inter-class Hamming distance is as large as possible. Therefore, we introduce the following constraints for learning $\mathbf{B}^{\Phi}$:

$$\begin{aligned} d_{\mathcal{H}}(\mathbf{b}_i^{\Phi}, \mathbf{b}_j^{\Phi}) &\leq \mu - \tau, \text{ if } \widetilde{S}_{ij}^{\Phi} = 1 \\ d_{\mathcal{H}}(\mathbf{b}_i^{\Phi}, \mathbf{b}_j^{\Phi}) &\geq \mu + \tau, \text{ if } \widetilde{S}_{ij}^{\Phi} = -1 \end{aligned} \tag{3.1}$$

where μ and τ are two hyper-parameters that are utilized to control intra-class compactness and inter-class separability, respectively. These two constraints can be modeled by the hinge loss function $g(z) = \max(z, 0)$. For example, if $\widetilde{S}_{ij}^{\Phi} = 1$, the first constraint in Eq (3.1) can be transformed into a minimum problem, i.e., $\arg\min_{\mathbf{b}} \max(d_{\mathcal{H}}(\mathbf{b}_i^{\Phi}, \mathbf{b}_j^{\Phi}) - \mu + \tau, 0)$. However, the hinge loss function is not differentiable. Following [22], we choose a generalized logistic loss function $f(z)$ to smoothly approximate the hinge loss function. Then, the objective to learn $\mathbf{b}_i^{\Phi}$ and $\mathbf{b}_j^{\Phi}$ can be formulated as

$$\mathcal{L}_1(h(\mathbf{x}_i^{\Phi}), h(\mathbf{x}_j^{\Phi}); \widetilde{S}_{ij}^{\Phi}) = \widetilde{S}_{ij}^{\Phi} f(\varphi_{ij}) + (1 - \widetilde{S}_{ij}^{\Phi}) f(-\varphi_{ij} + 2\tau), \tag{3.2}$$

where $\varphi_{ij} = d_{\mathcal{H}}(h(\mathbf{x}_i^{\Phi}), h(\mathbf{x}_j^{\Phi})) - \mu + \tau$.

In addition, we develop a label fitting method based on the discrete class proxies $\mathbf{P} \in \{-1, 1\}^{l \times q}$ and binary codes $\mathbf{B}^{\Delta}$. In order to make the binary codes $\mathbf{B}^{\Delta}$ carry sufficient supervised information, we expect that the intra-class similarities between the proxies and hash codes are maximized and the inter-class similarities are minimized. In other words, we expect semantic similar hash codes to share the same class proxy and semantic dissimilar hash codes to share different class proxies. More specifically, assume we have one discrete class proxy $\mathbf{p}_j \in \{-1, 1\}^l$ and one discrete binary code $\mathbf{b}_k^{\Delta} \in \mathbf{B}^{\Delta}$. We expect their similarity to approach 1 if they are semantic similar and -1 otherwise. Then, the objective to learn

b^{Δ}_{mk} can be formulated as

$$\mathcal{L}_2(\mathbf{P}, \mathbf{B}^{\Delta}) = \sum_{j=1}^{l}\sum_{k=1}^{n}(Y_{jk} - \mathbf{p}_j^T\mathbf{b}_k^{\Delta})^2 \tag{3.3}$$

We can find that the supervision information is directly enforced on $\mathbf{B}^{\Delta}$ through the proxy $\mathbf{P}$. Therefore, the semantic similarity can be expected to be well preserved between the proxy and binary codes. The whole objective function can be written as

$$\begin{aligned}\min_{h,\mathbf{B}^{\Delta}} \mathcal{L}(h, \mathbf{B}^{\Delta}) &= \sum_{i,j\in\Phi}\mathcal{L}_1(h(\mathbf{x}_i), h(\mathbf{x}_j); S_{ij}) + \alpha\frac{|\Phi|}{|\Delta|}\mathcal{L}_2(\mathbf{P}, \mathbf{B}^{\Delta}) \\ &= \sum_{i,j=1}^{|\Phi|}(\widetilde{S}_{ij}^{\Phi} f(\varphi_{ij}^{\Phi}) + (1 - \widetilde{S}_{ij}^{\Phi}) f(-\varphi_{ij}^{\Phi} + 2\tau)) + \alpha\frac{|\Phi|}{|\Delta|}\sum_{j=1}^{l}\sum_{k=1}^{n}(Y_{jk} - \mathbf{p}_j^T\mathbf{b}_k^{\Delta})^2,\end{aligned} \tag{3.4}$$

where the regularization $\mathcal{L}_3$ is not contained and will be introduced in the following subsection. In the following, we will use this objective function to guide the feature learning procedure and the hash coding procedure simultaneously.

3.2. Model optimization

In this subsection, we will design an efficient alternating method for solving the discrete optimization problem (3.4). Specifically, we optimize one of the variables of $\mathbf{B}^{\Delta}$, $\mathbf{P}$ and $h(\mathbf{X}^{\Phi})$ at a time with the other variable fixed. The detailed optimization process is presented as follows:

3.2.1. Fix $\mathbf{B}^{\Delta}$, $\mathbf{P}$ and update $\mathbf{Z}^{\Phi}$

The derivative of $h(\mathbf{X}^{\Phi}) = \text{sign}(F(\mathbf{X}^{\Phi}; \Theta))$ is zero everywhere, which leads to the vanishing gradient problem. Therefore, the back-propagation (BP) algorithm is unavailable to update Θ directly. In this paper, we add the $\tanh(\cdot)$ function as the activation function followed by the last layer to approximate the sign function. Inspired by [11, 13], we set $h(\mathbf{X}^{\Phi}) = F(\mathbf{X}^{\Phi}; \Theta)$ to integrate the above feature learning part and the loss part into an end-to-end framework. For brevity, the network output vector $F(\mathbf{x}_j^{\Phi}; \Theta)$ is denoted as $\mathbf{z}_j^{\Phi}$ and the m-th dimension output value $h_m(\mathbf{x}_j^{\Phi})$ is represented as z_{mj}^{Φ} ($z_{mj}^{\Phi} \in (-1, 1)$). In addition, we add a regularization term $\mathcal{L}_3(h(\mathbf{X}^{\Phi})) = \|\mathbf{Z}^{\Phi} - \mathbf{B}^{\Phi}\|_F^2$ to penalize the large deviation from the discrete binary codes for reducing the quantization errors. Then we can rewrite the problem (3.4) as

$$\min_{\mathbf{Z}^{\Phi}} \mathcal{L}(\mathbf{Z}^{\Phi}) = \sum_{i,j=1}^{|\Phi|}(\widetilde{S}_{ij}^{\Phi} f(\varphi_{ij}^{\Phi}) + (1 - \widetilde{S}_{ij}^{\Phi}) f(-\varphi_{ij}^{\Phi} + 2\tau)) + \alpha\frac{|\Phi|}{|\Delta|}\sum_{j=1}^{l}\sum_{k=1}^{n}(Y_{jk} - \mathbf{p}_j^T\mathbf{b}_k^{\Delta})^2 + \beta\|\mathbf{Z}^{\Phi} - \mathbf{B}^{\Phi}\|_F^2. \tag{3.5}$$

where $\varphi_{ij}^{\Phi} = (\frac{q}{2} - \mu + \tau) - \frac{1}{2}\mathbf{z}_i^{\Phi^T}\mathbf{z}_j^{\Phi}$. The derivatives of the problem (3.5) with respect to $\mathbf{z}_j^{\Phi}$ are given by:

$$\frac{\partial\mathcal{L}}{\partial\mathbf{z}_j^{\Phi}} = 2\sum_{i=1}^{|\Phi|}(\widetilde{S}_{ij}^{\Phi}\frac{\partial f(\varphi_{ij}^{\Phi})}{\partial\mathbf{z}_j^{\Phi}} + (1 - \widetilde{S}_{ij}^{\Phi})\frac{\partial f(-\varphi_{ij}^{\Phi} + 2\tau)}{\partial\mathbf{z}_j^{\Phi}}) + \beta(2\mathbf{z}_j^{\Phi} - \mathbf{b}_j^{\Phi}). \tag{3.6}$$

We have $\frac{\partial f(\varphi_{ij}^{\Phi})}{\partial \varphi_{ij}^{\Phi}} = \sigma(\rho\varphi_{ij}^{\Phi})$ and $\frac{\partial \varphi_{ij}^{\Phi}}{\partial \mathbf{z}_j^{\Phi}} = -\frac{1}{2}\mathbf{z}_i^{\Phi}$ where $\sigma(x) = \frac{1}{1+\exp(-x)}$ is the sigmoid function. Using the chain rule, we compute the gradient as follows:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}_j^{\Phi}} = \sum_{i=1}^{|\Phi|}(-\widetilde{S}_{ij}^{\Phi}\sigma(\rho\varphi_{ij}^{\Phi}) + (1-\widetilde{S}_{ij}^{\Phi})\sigma(-\rho\varphi_{ij}^{\Phi} + 2\rho\tau))\mathbf{z}_i^{\Phi} + \beta(2\mathbf{z}_j^{\Phi} - \mathbf{b}_j^{\Phi}). \tag{3.7}$$

3.2.2. Fix $\mathbf{Z}^{\Phi}$, $\mathbf{P}$ and update $\mathbf{B}^{\Delta}$

We fix $\mathbf{Z}^{\Phi}$ and $\mathbf{P}$ and rewrite the problem (3.4) as

$$\begin{aligned}\min_{\mathbf{B}^{\Delta}} \mathcal{L}(\mathbf{B}^{\Delta}) &= \sum_{j=1}^{l}\sum_{k=1}^{n}(Y_{jk} - \mathbf{p}_j^T\mathbf{b}_k^{\Delta})^2 + \beta\sum_{k\in\Phi}\|\mathbf{z}_k^{\Delta} - \mathbf{b}_k^{\Delta}\|^2 \\ &= \sum_{j=1}^{l}\sum_{k=1}^{n}(Y_{jk}^2 - 2Y_{jk}\mathbf{p}_j^T\mathbf{b}_k^{\Delta} + \mathbf{p}_j^T\mathbf{b}_k^{\Delta}(\mathbf{b}_k^{\Delta})^T\mathbf{p}_j) + \beta\sum_{k\in\Phi}(\mathbf{z}_k^{\Delta}(\mathbf{z}_k^{\Delta})^T - 2(\mathbf{z}_k^{\Delta})^T\mathbf{b}_k^{\Delta} + \mathbf{b}_k^{\Delta}(\mathbf{b}_k^{\Delta})^T) \\ &= const - 2\sum_{j=1}^{l}\sum_{k=1}^{n}Y_{jk}\mathbf{p}_j^T\mathbf{b}_k^{\Delta} - 2\beta\sum_{k\in\Phi}(\mathbf{z}_k^{\Delta})^T\mathbf{b}_k^{\Delta},\end{aligned} \tag{3.8}$$

Finally, we have the solution:

$$\mathbf{b}_k^{\Delta} = \text{sign}(\sum_{j=1}^{l}Y_{jk}\mathbf{p}_j + \beta\mathbf{z}_k^{\Delta}). \tag{3.9}$$

3.2.3. Fix $\mathbf{Z}^{\Phi}$, $\mathbf{B}^{\Delta}$ and update $\mathbf{P}$

We fix $\mathbf{Z}^{\Phi}$ and $\mathbf{B}$ and rewrite the problem (3.4) as

$$\begin{aligned}\min_{\mathbf{P}} \mathcal{L}(\mathbf{P}) &= \sum_{j=1}^{l}\sum_{k=1}^{n}(Y_{jk} - \mathbf{p}_j^T\mathbf{b}_k^{\Delta})^2 \\ &= \sum_{j=1}^{l}\sum_{k=1}^{n}(Y_{jk}^2 - 2Y_{jk}\mathbf{p}_j^T\mathbf{b}_k^{\Delta} + \mathbf{p}_j^T(\mathbf{b}_k^{\Delta})^T\mathbf{b}_k^{\Delta}\mathbf{p}_j) \\ &= const - 2\sum_{j=1}^{l}\sum_{k=1}^{n}Y_{jk}(\mathbf{b}_k^{\Delta})^T\mathbf{p}_j,\end{aligned} \tag{3.10}$$

Finally, we have the solution:

$$\mathbf{p}_j = \text{sign}(\sum_{j=1}^{l}Y_{jk}\mathbf{b}_k^{\Delta}). \tag{3.11}$$

3.3. Out-of-sample extension

Given an unseen data point in the test set, we can utilize the learnt deep hashing model to determine its hash code. More specifically, assume that we have an unseen instance $\mathbf{x}_u \notin \mathbf{X}$, and its hash code $\mathbf{b}_u \in \{-1, 1\}^q$ can be determined through forward propagation as follows:

$$\mathbf{b}_u = h(\mathbf{x}_u) = \text{sign}(F(\mathbf{x}_u, \Theta)), \tag{3.12}$$

where Θ denotes the set of network parameters.

4. Experiments

4.1. Experimental settings

4.1.1. Datasets

To verify the effectiveness of the proposed method, three widely used datasets including CIFAR-10, SVHN [23] and NUS-WIDE [24] are adopted to evaluate the proposed method and other baselines.

- **CIFAR-10** consists of 60,000 32×32 color images, which is attributed to a single-label dataset with 10 classes. Each class contains 6000 images. Following the setting in [11], we randomly take 100 images per class as the query set, and the remaining images form the database.
- **SVHN** consists of 630,420 digit images including 73,257 digits for training, 26,032 digits for test and 531,131 additional samples. It is also a single-label dataset, and the images are divided into ten classes with one for each digit. We randomly take 100 images per class as the query set and the whole training set forms the database.
- **NUS-WIDE** consists of 269,648 images, which is attributed to a multi-label dataset. Each image is associated with some of the 81 categories. Following the setting in [11, 15], we only use the images that belong to the 21 most frequent labels. Then it contains at least 5,000 images for each class. We randomly sample 2100 images (100 images per class) as the query set, and the remaining images form the dataset.

For single-label CIFAR-10 and SVHN datasets, two images sharing one common label will be considered to be a similar pair. Otherwise, they are treated as a dissimilar pair. For the multi-label dataset NUS-WIDE, two images sharing at least one label will be considered to be semantically similar. Otherwise, they are considered to be semantically dissimilar. Following the setting in [11], we randomly select 500 images per class from the database as the training set for all the datasets. In order to reduce randomness, 5 database/query splits are randomly generated. Finally, the performances across all these splits are averaged for each method. The choice of parameters α, β, and τ depends on cross-validation. The proposed model is implemented using PyTorch on an NVIDIA Titan-X GPU.

4.1.2. Evaluation metric

To evaluate the hashing-based retrieval performance, we adopt the criteria of Hamming ranking, which is widely used in the literature [25]. Given a query image, all images in the database are ranked according to their Hamming distances to the query image, and the desired neighbors from the top of the ranked list are returned. The retrieval performance is quantitatively evaluated by mean average precision (MAP).

$$precision@top(N) = \frac{N_t}{N} \tag{4.1}$$

where N_t denotes the number of retrieved relevant images in top N retrieved images. Average precision (AP) of top N retrieved images is defined as:

$$AP = \frac{1}{N_t} \sum_{i=1}^{N} precision@top(i) \times \delta(i) \tag{4.2}$$

where $\delta(i)$ is an indicator function, i.e., if the item at rank i is a relevant neighbor of the given query, $\delta(i) = 1$, otherwise $\delta(i) = 0$. The MAP can be obtained by averaging AP over all queries.

4.1.3. Implementation details

We first resize each input image to 224×224 using bilinear interpolation for three datasets, and the raw image pixels are directly used as network inputs. The mini-batch size is fixed to be 128. The learning rate is tuned from 10^{-2} to 10^{-6} by cross validation. To avoid overfitting, the weight decay is set to 5×10^{-4}. The maximum number of iterations is set to 150. The constant factor Δ is set to 1.01. Furthermore, for the following experiments, the parameters μ and β are empirically set to $\frac{q}{2}$ and 1, respectively. α and τ are selected from the sets of $\{1, 10^{-1}, 10^{-2}, 10^{-3}\}$ and $\{0, 2, 4, 6\}$ by cross-validation on the training set, respectively. We compare the proposed method with four non-deep learning methods, namely LFH [26], FastH [27], SDH [5], and COSDISH [21], and three deep learning methods, namely DPSH [11], DHN [13], and HashNet [15]. For the non-deep learning methods, we apply the CNN-F model pre-trained on ImageNet to extract a 4096-dimensional global CNN features as image representations for fair comparison. Following the setting in FastH [27], the boosted decision trees are chosen for out-of-sample extension by LFH [26], FastH [27], and COSDISH [21].

Table 2. Results in terms of MAP of all the compared methods on the three datasets with different code lengths.

Method	CIFAR-10			SVHN			NUS-WIDE		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
LFH [26]	0.4979	0.6048	0.6875	0.2914	0.3798	0.4399	0.6415	0.6661	0.6904
FastH [27]	0.6415	0.6886	0.7177	0.3648	0.4137	0.4542	0.6027	0.6686	0.6877
SDH [5]	0.6189	0.6681	0.6775	0.3273	0.3507	0.3824	0.5471	0.5585	0.5853
COSDISH [21]	0.6370	0.6881	0.7165	0.35270	0.4101	0.4576	0.6015	0.6654	0.6891
DPSH [11]	0.7091	0.7395	0.7551	0.6090	0.6689	0.7086	0.6563	0.6751	0.6888
DHN [13]	0.7289	0.7449	0.7188	0.5895	0.6314	0.6439	0.6675	0.6876	0.6890
HashNet [15]	0.7413	0.7715	0.7663	0.7100	0.7391	0.7220	**0.6794**	0.6992	0.7098
Ours	**0.7934**	**0.8127**	**0.8258**	**0.7312**	**0.8268**	**0.8387**	0.6768	**0.7144**	**0.7325**

4.2. Experimental results

The retrieval performances on CIFAR-10 in terms of MAP with respect to different code lengths are shown in the second to the fourth columns of Table 2. We can find that our method outperforms all other baseline methods by a large margin. The improvement can be attributed to the introduction of metric learning and label fitting methods and the discrete optimization process for the proposed loss. We also notice that the deep supervised hashing methods DPSH [11], DHN [13], and HashNet [15] and our method outperform the traditional supervised hashing methods LFH [26], FastH [27], SDH [5], and COSDISH [21] with CNN features. These results demonstrate that end-to-end deep learning architecture is more compatible with hashing procedures.

The image retrieval results (MAP) of our proposed method and baseline methods on the SVHN dataset for different code lengths are reported in the fifth to the seventh columns of Table 2. We can

observe that the proposed method performs better than the baseline methods. COSDISH [21] and FastH [27] obtain comparable performance. The reason can be that they both try to learn hash codes by approximating the semantic similarity with Hamming affinity. In addition, for solving the resulting BQP problem, FastH [27] uses a Block Graph-Cut method, while COSDISH [21] transforms this problem into an equivalent clustering problem. The results of all compared methods in terms of MAP on the NUS-WIDE dataset are reported in the eighth to the tenth columns of Table 2. The performance of the proposed method is slightly lower than that of HashNet [15] with the code length of 16 bits on the NUS-WIDE dataset. The main reason can be attributed to the multi-label properties of the NUS-WIDE, which brings a slight challenge to fit the semantic labels in lower bits. Except for the 16 bits, our method achieves the best performance.

Table 3 demonstrates the comparison of the training time (in seconds) cost in one iteration w.r.t. different number of bits on the NUS-WIDE dataset. From Table 3, we can see that the training time of the proposed method is comparable with other baseline methods. In Figure 2, we demonstrate some retrieval examples of top 10 retrieved images returned by the proposed method for the CIFAR-10 dataset. It can be seen that the proposed method can return relevant results.

Table 3. Comparison of the training time (in seconds) cost in one iteration w.r.t. different number of bits on NUS-WIDE dataset.

Methods	**NUS-WIDE**		
	16 bits	32 bits	64 bits
DPSH	109.89	128.60	150.79
HashNet	105.62	122.89	146.45
Ours	118.38	137.09	153.53

4.3. Sensitivity to parameters

In this subsection, we analyze the effects of the parameters α, β, and τ on the algorithm performance of our method. In the following experiments, the code length is fixed at 64 bits. In addition, the MAP score is used to reflect the variation in performance with different parameter values. The impacts of different α, β, and τ on the CIFAR-10, SVHN, and NUS-WIDE datasets are shown in Figure 3. The parameter α controls the balance between the label fitting term and the other terms. The parameter τ controls the margin between inter-class hash codes in the metric learning term. The parameter β controls the quantization errors. As we can see, the proposed method achieves the best performance around $\alpha = 10^{-2}$, $\beta = 1$, and $\tau = 2$ on CIFAR-10 and SVHN datasets, and $\alpha = 10^{-2}$, $\beta = 1$, and $\tau = 4$ for the multi-label dataset NUS-WIDE. In addition, as τ increases, the experimental performances first increase and then decrease slightly. The reason can be that a proper margin can improve the discriminative ability of hash codes, while a too large margin may introduce a lot of hard pairwise negatives for training the model. As α increases, the experimental performances first increase and then decrease significantly. The reason can be attributed to the imbalance of the components in the proposed model for a too large α.

4.4. Ablation study

The proposed method consists of three components: metric learning $\mathcal{L}_1$, label fitting $\mathcal{L}_2$, and the regular term $\mathcal{L}_3$. In this subsection, we study the contribution of different components to the image

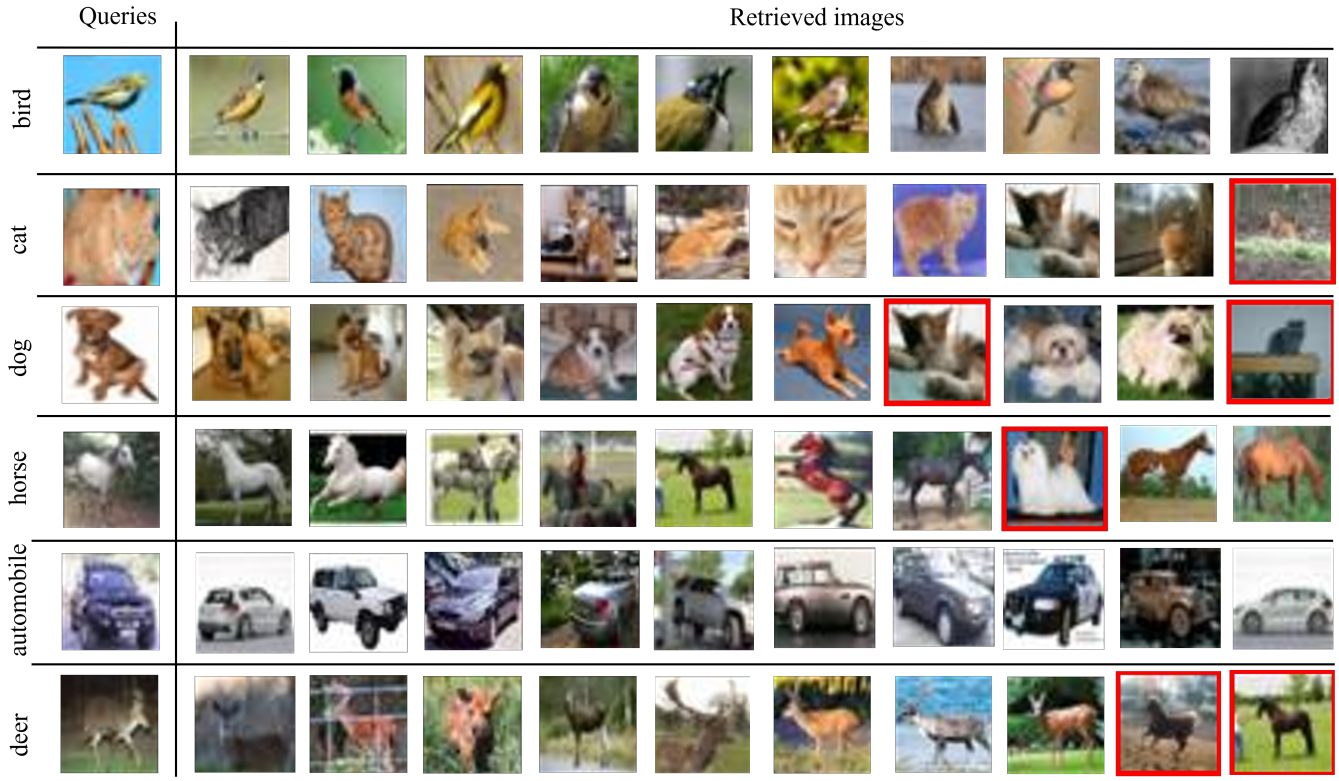
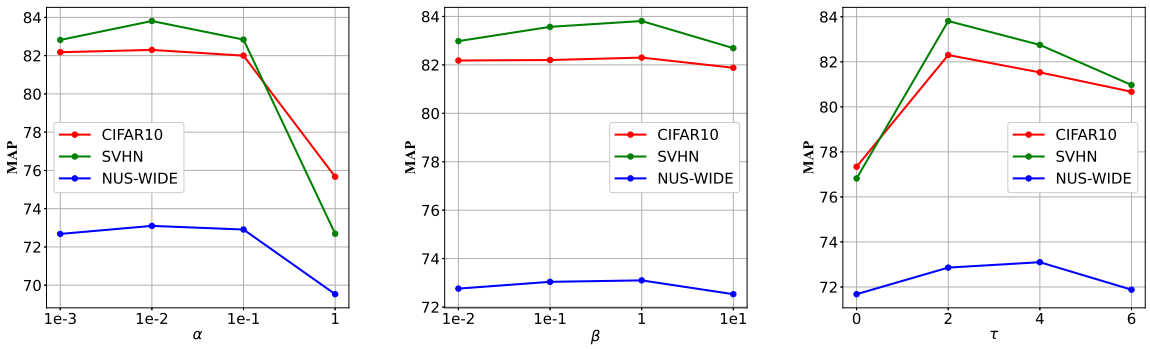


Figure 2. Some examples of top 10 retrieved images returned by the proposed method for CIFAR-10 dataset. Images with a red box denote incorrect retrieval samples.



(a) Impact of α **(b)** Impact of β **(c)** Impact of τ

Figure 3. Impact of different α, β, and τ on the CIFAR-10, SVHN, and NUS-WIDE datasets at 64 bits.

retrieval performance. Since $\mathcal{L}_2$ and $\mathcal{L}_3$ are mutually dependent, they should be retained or removed simultaneously during the ablation study. Table 4 illustrates the experimental results of different components and their combinations. From Table 4, we can see that the metric learning $\mathcal{L}_1$ can achieve promising image retrieval performance compared with other deep supervised hashing methods. A reasonable explanation is that the introduced proper margin can improve the discriminative ability of the learned hash codes. From Table 4, we can observe that our method achieves the best performance when it combines these three components.

Table 4. The ablation study on CIFAR-10, SVHN, and NUS-WIDE datasets using different combinations of components.

Components	CIFAR-10			SVHN			NUS-WIDE		
	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
$\mathcal{L}_1$	0.7761	0.8069	0.8218	0.6954	0.8072	0.8282	0.6734	0.7076	0.7268
$\mathcal{L}_1 + \mathcal{L}_2 + \mathcal{L}_3$	**0.7934**	**0.8127**	**0.8258**	**0.7312**	**0.8268**	**0.8387**	**0.6768**	**0.7144**	**0.7325**

5. Conclusions

In this paper, we propose a new deep supervised hashing method for large-scale content-based image retrieval. Specifically, we utilize a metric learning strategy to minimize the intra-class Hamming distance and maximize inter-class Hamming distance as much as possible. Then, a novel label fitting approach is developed to generate discrete proxies and hash codes simultaneously. Finally, an effective alternating optimization method is designed to solve the discrete optimization problem. Experimental results on various datasets demonstrate that the proposed method can achieve promising retrieval performance for large-scale content-based image retrieval systems.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

We are appreciated for all of the anonymous reviewers efforts for the improvement of the whole research.

Conflict of interest

The authors declare there is no conflicts of interest.

References

1. P. Indyk, R. Motwani, Approximate nearest neighbors: Towards removing the curse of dimensionality, *ACM Symp. Theory Comput.*, (1998), 604–613. https://doi.org/10.1145/276698.276876

2. C. S. Anan, R. I. Hartley, Optimised kd-trees for fast image descriptor matching, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2008), 1–12. https://doi.org/10.1109/CVPR.2008.4587638

3. A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in *International Conference on Very Large Data Bases*, (1999), 518–529. https://dl.acm.org/doi/10.1145/3701716.3717581

4. Y. C. Gong, S. Lazebnik, Iterative quantization: A procrustean approach to learning binary codes, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2011), 817–8294. https://doi.org/10.1109/CVPR.2011.5995432

5. F. M. Shen, C. H. Shen, W. Liu, H. T. Shen, Supervised discrete hashing, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2015), 37–45. https://doi.org/10.1109/CVPR.2015.7298598

6. J. Wang, W. Liu, S. Kumar, S. F. Chang, Learning to hash for indexing big data—A survey, *Proc. IEEE*, **104** (2016), 34–57. https://doi.org/10.1109/JPROC.2015.2487976

7. R. K. Xia, Y. Pan, H. J. Lai, C. Liu, S. C. Yan, Supervised hashing for image retrieval via image representation learning, in *AAAI Conference on Artificial Intelligence*, (2014), 2156–2162. https://doi.org/10.1609/AAAI.V28I1.8952

8. H. J. Lai, Y. Pan, Y. Liu, S. C. Yan, Simultaneous feature learning and hash coding with deep neural networks, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2015), 3270–3278. https://doi.org/10.1109/CVPR.2015.7298947

9. F. Zhao, Y. Z. Huang, L. Wang, T. N. Tan, Deep semantic ranking based hashing for multi-label image retrieval, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2015), 1556–1564. https://doi.org/10.1109/CVPR.2015.7298763

10. R. M. Zhang, L. Lin, R. Zhang, W. M. Zuo, L. Zhang, Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification, *IEEE Trans. Image Process.*, 24(2015), 4766–4779. https://doi.org/10.1109/TIP.2015.2467315

11. W. J. Li, S. Wang, W. C. Kang, Feature learning based deep supervised hashing with pairwise labels, in *International Joint Conference on Artificial Intelligence*, (2016), 1711–1717. https://dl.acm.org/doi/10.5555/3060832.3060860

12. H. M. Liu, R. P. Wang, S. G. Shan, X. L. Chen, Deep supervised hashing for fast image retrieval, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2016), 2064–2072. https://doi.org/10.1109/CVPR.2016.227

13. H. Zhu, M. S. Long, J. M. Wang, Y. Cao, Deep hashing network for efficient similarity retrieval, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2016), 2415–2421. https://doi.org/10.1609/AAAI.V30I1.10235

14. Y. Cao, M. S. Long, J. M. Wang, H. Zhu, Q. F. Wen, Deep quantization network for efficient image retrieval, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2016), 3457–3463. https://doi.org/10.1609/AAAI.V30I1.10455

15. Z. J. Cao, M. S. Long, J. M. Wang, P. S. Yu, Hashnet: Deep learning to hash by continuation, in *Proceedings of the IEEE International Conference on Computer Vision*, (2017), 5609–5618. https://doi.org/10.1109/ICCV.2017.598

16. Z. D. Chen, L. J. Zhao, Z. C. Zhang, X. Luo, X. S. Xu, Characteristics matching based hash codes generation for efficient fine-grained image retrieval, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2024), 17273–17281. https://doi.org/10.1109/CVPR52733.2024.01635

17. L. D. Wang, Y. Pan, C. Liu, H.J. Lai, J. Yin, Y. Liu, Deep hashing with minimal-distance-separated hash centers, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2023), 23455–23464. https://doi.org/10.1109/CVPR52729.2023.02246

18. L. Yuan, T. Wang, X. P. Zhang, F. E. H. Tay, Z. Q. Jie, Y. H. Tian, et. al., Learnable central similarity quantization for efficient image and video retrieval, *IEEE Trans. Neural Networks Learn. Syst.*, **35** (2024), 18717–18730. https://doi.org/10.1109/TNNLS.2023.3321148

19. W. C. Kang, W. J. Li, Z. H. Zhou, Column sampling based discrete supervised hashing, in *Proceedings of the AAAI Conference on Artificial Intelligence*, (2016), 1230–1236. https://doi.org/10.1609/aaai.v30i1.10176

20. K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: Delving deep into convolutional nets, preprint, arXiv:1405.3531.

21. A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, *Adv. Neural Inf. Process. Syst.*, (2012), 1106–1114. https://doi.org/10.1145/3065386

22. V. E. Liong, J. W. Lu, Deep coupled metric learning for cross-modal matching, *IEEE Trans. Multimedia*, **19** (2017), 1234– 1244. https://doi.org/10.1109/TMM.2016.264618

23. Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A. Y. Ng, Reading digits in natural images with unsupervised feature learning, in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, (2011), 12–17. https://doi.org/10.1007/978-3-319-10593-2_33

24. T. S. Chua, J. H. Tang, R. C. Hong, H. J. Li, Z. P. Luo, Y. T. Zheng, NUS-WIDE: a real-world web image database from national university of singapore, in *ACM International Conference on Image and Video Retrieval*, (2009), 1–9. https://doi.org/10.1145/1646396.1646452

25. J. Wang, S. Kumar, S. F. Chang, Semi-supervised hashing for large-scale search, *IEEE Trans. Pattern Anal. Mach. Intell.*, **34** (2012), 2393–2406. https://doi.org/10.1109/TPAMI.2012.48

26. P. C. Zhang, W. Zhang, W. J. Li, M. Y. Guo, Supervised hashing with latent factor models, in *ACM SIGIR Conference on Research and Development in Information Retrieval*, (2014), 173–182. https://doi.org/10.1145/2600428.2609600

27. G. S. Lin, C. H. Shen, Q. F. Shi, A. V. D. Hengel, D. Suter, Fast supervised hashing with decision trees for high-dimensional data, in *IEEE Conference on Computer Vision and Pattern Recognition*, (2014), 1971–1978. https://doi.org/10.1109/CVPR.2014.253