International Journal of Software and Informatics, ISSN 1673-7288
http://www.ijsi.org, ijsi@iscas.ac.cn, +86-10-62661048
IJSI, 2021, 11(1): 55–67, doi: 10.21655/ijsi.1673-7288.00240

Research Article

Local Semantic Structure Captured and Instance Discriminated by Unsupervised Hashing

Changsheng Li (李长升) [1], Qixing Min (闵齐星) [2], Yurong Cheng (成雨蓉) [1], Ye Yuan (袁野) [1], Guoren Wang (王国仁) [1]

[1] (School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China)

[2] (School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China)

Corresponding author: Changsheng Li, changshengli507@163.com

Abstract Recently, unsupervised Hashing has attracted much attention in the machine learning and information retrieval communities, due to its low storage and high search efficiency. Most of existing unsupervised Hashing methods rely on the local semantic structure of the data as the guiding information, requiring to preserve such semantic structure in the Hamming space. Thus, how to precisely represent the local structure of the data and Hashing codes becomes the key point to success. This study proposes a novel Hashing method based on self-supervised learning. Specifically, it is proposed to utilize the contrastive learning to acquire a compact and accurate feature representation for each sample, and then a semantic structure matrix can be constructed for representing the similarity between samples. Meanwhile, a new loss function is proposed to preserve the semantic information and improve the discriminative ability in the Hamming space, by the spirit of the instance discrimination method proposed recently. The proposed framework is end-to-end trainable. Extensive experiments on two large-scale image retrieval datasets show that the proposed method can significantly outperform current state-of-the-art methods.

Keywords unsupervised Hashing; contrastive learning; instance discrimination; local semantic structure

Citation Li CS, MIN QX, Cheng YR, Yuan Y, Wang GR. Local semantic structure captured and instance discriminated by unsupervised hashing. *International Journal of Software and Informatics*, 2021, 11(1): 55–67. http://www.ijsi.org/1673-7288/00240.htm.

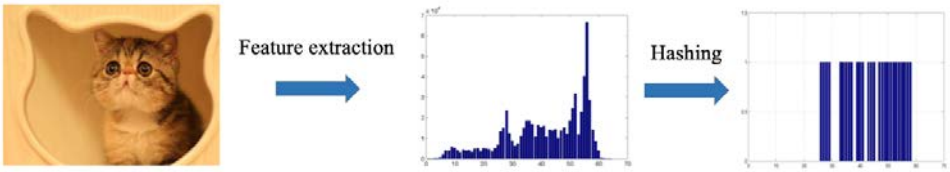
With the rapid development of the Internet and the explosive growth of data (such as pictures, videos, documents), how to quickly retrieve the information users need has become one of the hot issues in the academic and industrial circles. As one of the most efficient methods for large-scale information retrieval, Hashing has developed by leaps and bounds in recent years. In principle, it

This is the English version of the Chinese article “捕获局部语义结构和实例辨别的无监督哈希. 软件学报, 2021, 32(3): 742−752. doi: 10.13328/j.cnki.jos.006178”.

Funding items: National Natural Science Foundation of China (61806044, U2001211, 61932004, 61732003); Research Fund Program of Beijing Institute of Technology for Young Scholars (3070012222010)

Received 2020-07-02; Revised 2020-09-03; Accepted 2020-11-06; IJSI published online 2021-03-31

usually maps the data (such as images, videos, and texts) in a high-dimensional continuous space to a low-dimensional binary space (i.e., Hamming space), as shown in Figure 1. During the mapping, it is expected to maintain the information concerning the original space in the Hamming space. Representing the data features with binary codes, Hashing can greatly reduce the storage cost and computational complexity and thus retrieve large-scale data sets quickly. Therefore, it is deemed as a new technology to support efficient feature learning in large-scale data retrieval. Because of its great potential, Hashing has been widely applied to various tasks, including cross-media retrieval[1], recommendation systems[2], and copy detection[3].



(a) Original image (b) Color histogram (c) Hash code

Figure 1 Brief introduction of Hashing

Most of the existing Hashing methods are independent of data. For example, the typical Locality Sensitive Hashing (LSH)[4] attempts to generate embedded representation by random mapping. One advantage of such a technology is that in extreme cases, with the increase in Hash code bits, random mapping can keep the distance between inputs. Considering that the generation of a Hash function by this method does not depend on the dataset itself, the obtained function may not be globally optimal. In recent years, data-dependent Hashing has gained increasing attention and developed rapidly[5]. By modeling based on target dataset learning, it produces more accurate and concise Hash codes. Since the data-dependent methods often yield satisfactory results, various Hashing methods have been proposed, which can be roughly classified into supervised methods[6–8], semi-supervised methods[9,10] and unsupervised methods[11–17] according to whether supervised information is involved. Supervised Hashing usually applies supervised information (such as label information) to Hashing, where the supervised information includes the label information of a single sample, paired samples and sequences. The representatives include Supervised Discrete Hashing (SDH)[18], Fast supervised Hashing (FastH)[6], Ranking-based Supervised Hashing (RSH)[19] and Convolutional Neural Network Hashing (CNNH)[20]. However, such methods have some problems. To be specific, they usually require the Hash function to possess strong discriminability, otherwise the performance of the model fails to be guaranteed. Moreover, due to complex data in the actual scenes, more codes are needed to ensure the accuracy of the model, which lifts the storage burden. Wang *et al*.[9] put forward a semi-supervised Hashing method, where the learning of data pairs guaranteed similar Hash codes for similar data pairs but dissimilar codes for dissimilar pairs. Besides, the information entropy of Hash codes on unlabeled data should be maximized. Mu *et al*.[10] manually marked some semantically similar and dissimilar data pairs and adopted quadratic programming to obtain an accurate Hash function. Unsupervised Hashing methods do not use any supervised information but only the feature information of data for learning and training, represented by ITerative Quantization (ITQ)[11], Discrete Graph Hashing (DGH)[13], and Scalable Graph Hashing (SGH)[21]. Unsupervised Hashing is simple and easy to be implemented because it is free from label information in the learning process. However, it is challenging because it does not involve supervised information. Therefore, this paper mainly investigates unsupervised Hashing.

In the past few years, although substantial unsupervised Hashing methods have been put forward, they still have the following problems. (1) Because of no label information in the data, how to accurately construct the semantically similar structure between data is still a pending

question. (2) In the learning to Hash, most methods only try to maintain the semantic structure of data but ignore the discrimination of Hash codes. As we all know, the discrimination of data features is pivotal to downstream tasks[22], and thus how to improve the discrimination of Hash codes is worth exploring.

Inspired by instance discrimination[23], this paper proposes a novel Hashing method based on self-supervised learning. To be specific, it utilizes the contrastive learning to acquire a compact and accurate feature representation for each sample, and then a semantic structure matrix can be constructed for representing the similarity between samples. Meanwhile, a new loss function is proposed to preserve the semantic information and improve the discriminative ability in the Hamming space. In addition, a regular term is added to reduce the loss caused by relaxation. The framework proposed in this paper is end-to-end trainable and optimized by a standard back propagation algorithm. The image classification model VGG-F[24] is modified and trained accordingly. The feasibility and accuracy of the proposed method are compared with those of the state-of-the-art methods by retrieval experiments with FLICKR25K and NUSWIDE datasets.

Section 1 of this paper mainly introduces the existing algorithms of Hashing and their categories, the problems of unsupervised Hashing, and the main technical route of this paper. In Section 2, the performance improvement method of the model proposed in this paper is elaborated with respect to several aspects such as problem definition, network architecture, and loss function. Section 3 proves that this method can improve the retrieval accuracy of the model in the case of different Hash code lengths in two commonly used datasets. In Section 4, we summarize this paper and offer the prospects.

1 Related Work

Designing an efficient feature learning algorithm for large-scale datasets is of great significance for retrieval. During the construction of an efficient large-scale retrieval system, two main problems are always encountered: storage cost and retrieval speed. At present, because of the high-dimensional features of multimedia data such as texts, images and videos, retrieval methods are faced with severe challenges of “curse of feature dimensionality”, which dramatically increases the storage space and computational complexity, thereby affecting the performance of the retrieval system. For these reasons, learning to Hash has been put forward and become a research hotspot in the fields of information retrieval and machine learning. Ref. [5] analyzes the characteristics and structure of data and maps high-dimensional continuous data into Hash codes (i.e., binary strings) by machine learning. At the same time, the structure information in the original space is kept as much as possible in the Hamming space. Because of its binary representation, learning to Hash can significantly reduce the storage cost and computational complexity, thus effectively improving the retrieval efficiency. As the focus of this paper, unsupervised Hashing is reviewed.

Traditional unsupervised Hashing usually operates based on shallow structures. These methods usually regard feature learning and Hashing as two separate processes. ITQ[11], as a representative, attempts to reduce the dimension of the original dataset by Principal Component Analysis (PCA) and maps the data points in the dataset to the vertices of a binary hypercube, so as to minimize the corresponding quantization error. As a result, an accurate Hash code corresponding to the dataset is obtained. In recent years, as deep learning has achieved superior results in various visual tasks and machine learning, it has gradually been applied to Hashing, e.g., Semantic Hashing[25], Deep Auto-encoder Hashing[26] and Deep Binary descriptors (DeepBit)[27]. Semantic Hashing employs a pre-trained restricted Boltzmann mechanism to build an auto-encoder network, so as to produce effective Hash codes and accurately reconstruct the original input. In deep auto-encoder Hashing, a very deep auto-encoder is designed to map the original input into the Hamming space, and the reconstruction loss guides the learning to Hash. DeepBit integrates

feature learning and learning to Hash into a framework, achieving good results.

2 Methods

In this paper, a deep unsupervised Hashing method based on local semantic structure and instance discrimination is proposed. In the learning to Hash, improving the discriminability of Hash codes can enhance the expressiveness and retrieval of the model. This method mainly includes two parts. First, contrastive learning refines the local semantically similar structure, so that it can represent not only the semantic information of data but also the discrimination information. Secondly, a new target loss function is proposed. Contrastive learning enables the Hash codes to keep the semantic information and improves the discriminability of Hash codes. In the following part, we will elaborate the problem definition, network architecture, semantic structure matrix, and learning to Hash.

2.1 Problem definition

First of all, the representations of some main notations are listed in Table 1.

Table 1 Notations

Notation	Meaning
$\boldsymbol{A}$	Matrix
$\boldsymbol{a}_i$	The ith column of the matrix
$\boldsymbol{a}_{ij}$	Element in the ith row and the jth column of the matrix
$\|A\|_F$	Frobenius norm of the matrix
A^{T}	Transpose of $\boldsymbol{A}$
$\exp(\cdot)$	Exponent operation
$\boldsymbol{a}\cdot\boldsymbol{b}$	Inner product of two vectors
$\|\boldsymbol{A}\|$	Absolute value of each element
L	Length of a Hash code
N	Number of samples
d	Original feature dimension of a sample

Given a group of training data $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$, the goal of this paper is to learn a group of binary Hash codes:

$$B = [\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n] \in \{-1, 1\}^{L\times n}.$$

For this purpose, this paper intends to solve a group of effective Hash functions as follows:

$$\boldsymbol{b}_i = [h_1(\boldsymbol{x}_i), \ldots, h_L(\boldsymbol{x}_i)] = [\mathrm{sgn}(F(\boldsymbol{x}_i; \boldsymbol{W}_1)), \ldots, \mathrm{sgn}(F(\boldsymbol{x}_i; \boldsymbol{W}_L))] \tag{1}$$

where $\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L$ represent the parameters for model learning. $\mathrm{sgn}(\cdot)$ is a sign function defined as

$$\mathrm{sgn}(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ -1, & \text{if } x < 0 \end{cases} \tag{2}$$

Inspired by the outstanding performance of deep learning[27,28], this paper still takes deep neural networks as the basic framework to learn the Hash function in order to well map the original data to Hamming space. Although many previous methods try to keep the structural information of data in the Hamming space, they ignore the discriminability of Hash codes. Therefore, besides keeping the semantically similar structure of data in the Hamming space, this paper attempts to improve the discriminability of Hash codes.

2.2 Model learning

For the above goal, this paper proposes an architecture based on instance discrimination, as shown in Figure 2.

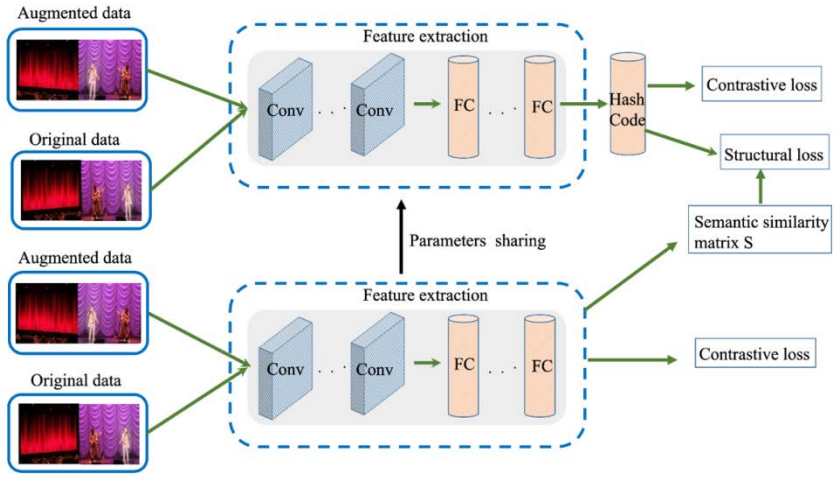


Figure 2 Architecture of the proposed method

The whole architecture is mainly divided into two parts: (1) construction of a semantic similarity matrix; (2) learning to Hash. Specifically, to construct a matrix $\boldsymbol{S}$, this paper firstly trains the target dataset by contrastive learning, so that the learned features are discriminative. After the model update, with the features in the middle layer of the network as the new feature representations of the data, the matrix $\boldsymbol{S}$ is constructed. Concerning the learning to Hash, we first optimize the network by the structural loss function and try to maintain the local semantic structure of data in the Hamming space. Then, the original data and augmented data form a training sample pair, and contrastive loss is used to enhance the discrimination of features.

During the construction of the matrix $\boldsymbol{S}$, the VGG-F[24] model is taken as the convolution backbone for feature extraction. As this paper studies the unsupervised Hashing, the label information of data is unavailable. By virtue of the self-supervised learning mechanism, this paper builds auxiliary tasks to learn the network. The following loss function is used in this paper:

$$L_c = -\log \frac{\exp(\boldsymbol{x}_i^* \cdot \boldsymbol{x}_i^+ / \tau)}{\sum_{j=1}^{m} \exp(\boldsymbol{x}_i^* \cdot \boldsymbol{x}_j^+ / \tau)} \tag{3}$$

where L_c is the batch loss of contrastive loss, i is i-th sample of the batch data, m is the batch size and τ is a hyperparameter. $\boldsymbol{x}_i^*$ and $\boldsymbol{x}_i^+$ are two augmented samples of $\boldsymbol{x}_i$, such as data augmentation of the original image through random rotation and adding noise. Equation (3) aims to form a positive sample pair with two augmented samples of same data and negative sample pairs with these augmented samples and those of other data to train a classifier. Consequently, the augmented samples of the same sample are classified into the same class. Through the above-mentioned auxiliary tasks, the features learned by the network are of certain discriminability. To avoid over-fitting, this paper does not use the original data directly but their augmented samples to update the network.

After the network update ceases, this paper employs the features of Layer fc-7 as the new feature representations and constructs the following similarity matrix:

$$\boldsymbol{S}_{ij} = \begin{cases} 1, & \text{if } \boldsymbol{x}_j \in \Theta_k(\boldsymbol{x}_i) \\ 0, & \text{if } \boldsymbol{x}_j \notin \Theta_k(\boldsymbol{x}_i) \cup \Omega_k(\boldsymbol{x}_i) \\ -1, & \text{if } \boldsymbol{x}_j \in \Omega_k(\boldsymbol{x}_i) \end{cases} \tag{4}$$

where $\Theta_k(\boldsymbol{x}_i)$ represents K nearest neighbors of point x_i, and $\Omega_k(\boldsymbol{x}_i)$ means K points farthest from x_i among all points. In Equation (4), if two points are neighbors, their semantic information is

considered similar. Hence, the distance between these two points in the Hamming space is small. If two points are far apart, then their semantic information is dissimilar and thus these two points are distant in the Hamming space. Algorithm 1 gives the concrete steps of constructing a structure matrix.

Algorithm 1. Construction of a semantic similarity matrix.

Input: Training data $\boldsymbol{X}$ and mini-batch of size m.

Output: Semantic similarity matrix $\boldsymbol{S}$.

1. Initialization: Initialize the parameters of VGG-F by the model pre-trained in the image net dataset;
2. Repeat until convergence;
3. Randomly select m samples from $\boldsymbol{X}$ to build a mini-batch;
4. Augment each sample in the mini-batch;
5. Calculate the loss function (3) by forward propagation;
6. Update the network parameters by a backpropagation algorithm;
7. Relying on the updated VGG-F, extract the features of Layer fc-7 as the new features of the dataset;
8. Construct a matrix $\boldsymbol{S}$ through Equation (4).

This paper proposes the following objective function to maintain the semantic structure of data in the Hamming space and improve the discriminability of Hashing features:

$$\min_{\boldsymbol{B}} L_b = -\log \frac{\exp(\boldsymbol{b}_i \cdot \boldsymbol{b}_i^+ / \tau)}{\sum_{j=1}^{m} \exp(\boldsymbol{b}_i \cdot \boldsymbol{b}_j^+ / \tau)} + \alpha \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} | \boldsymbol{S}_{ij} | (\boldsymbol{M}_{ij} - \boldsymbol{S}_{ij})^2 \tag{5}$$

where $\boldsymbol{b}_i$ and $\boldsymbol{b}_i^+$ respectively represent the Hash codes of $\boldsymbol{x}_i$ and its enhanced sample $\boldsymbol{x}_i^+$; $\boldsymbol{b}_i = \mathrm{sgn}(F(\boldsymbol{x}_i; \Phi))$; Φ refers to learnable parameters for network. Matrix $\boldsymbol{M}$ is defined as follows:

$$\boldsymbol{M}_{ij} = \frac{1}{L} \boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{b}_j, \boldsymbol{b}_i \in \{-1,1\}^L \tag{6}$$

In Equation (5), the purpose of the first item is to make the data sample and its augmented sample as close as possible in the Hamming space, so that the Hash codes are discriminative. The second item focuses on achieving consistent semantic structure of data in the Hamming space and continuous feature space. Jointly optimizing these two items can maintain the semantic structure of data and improve the data discrimination.

In Equation (5), the binary representation makes it very difficult to optimize the network. To update the network gradient effectively, this paper uses function tanh(·) instead of function sgn(·) to relax the objective function. Hence, the following objective function is proposed:

$$\left.\begin{aligned} &\min_{\boldsymbol{B}} L_b = -\log \frac{\exp(\boldsymbol{b}_i \cdot \boldsymbol{b}_i^+ / \tau)}{\sum_{j=1}^{m} \exp(\boldsymbol{b}_i \cdot \boldsymbol{b}_j^+ / \tau)} + \alpha \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} | \boldsymbol{S}_{ij} | (\boldsymbol{M}_{ij} - \boldsymbol{S}_{ij})^2 \\ &\text{s.t. } \boldsymbol{b}_i = \tanh(F(\boldsymbol{x}_i; \Phi)), \boldsymbol{M}_{ij} = \frac{1}{L} \boldsymbol{b}_i^{\mathrm{T}} \boldsymbol{b}_j \end{aligned}\right\} \tag{7}$$

In addition, this paper adds another regular term to make the Hash code as close to 1 or −1 as possible, minimizing the loss induced by the above relaxation. Thus, the following objective function is obtained:

$$\left.\begin{aligned}&\min_{\boldsymbol{B}} L_b = -\log\frac{\exp(\boldsymbol{b}_i\cdot\boldsymbol{b}_i^+/\tau)}{\sum_{j=1}^{m}\exp(\boldsymbol{b}_i\cdot\boldsymbol{b}_j^+/\tau)} + \alpha\frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}|\boldsymbol{S}_{ij}|(\boldsymbol{M}_{ij}-\boldsymbol{S}_{ij})^2 + \beta\sum_{i=1}^{L}(1-|\boldsymbol{b}_{ij}|)\\&\text{s.t. } \boldsymbol{b}_i = \tanh(F(\boldsymbol{x}_i;\Phi)), \boldsymbol{M}_{ij} = \frac{1}{L}\boldsymbol{b}_i^{\mathrm{T}}\boldsymbol{b}_j\end{aligned}\right\} \tag{8}$$

where $\alpha \geq 0$ and $\beta \geq 0$ are two hyper parameters.

The standard back propagation algorithm is adopted for gradient update so as to solve Equation (8). The whole training process is shown in Algorithm 2.

Algorithm 2. Training of Hash codes.

Input: Training data X, mini-batch of size m, and hyperparameters α and β.

Output: Parameters of neural network $\Phi = \{\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L\}$ and Hash codes for training data.

1. Initialization: Initialize the parameters of VGG-F by the fine-tuned model in the training dataset;
2. Repeat until convergence;
3. Select *m* samples randomly from $\boldsymbol{X}$ to build a mini-batch;
4. Augment each sample of the mini-batch;
5. Calculate the loss function (8) by forward propagation;
6. Update the network parameters by the back propagation algorithm;
7. On the basis of the updated VGG-F, the last layer of the network is extracted as the Hash code of the data.

After the network training is completed, the Hash code of any other point $\boldsymbol{x}_t$ that is not in the training set can be directly calculated by the following formula:

$$\boldsymbol{b}_t = \mathrm{sgn}(F(\boldsymbol{x}_t;\Phi)) \tag{9}$$

The mapping of Hash codes for arbitrary data points is shown in Algorithm 3.

Algorithm 3. Test of Hash codes.

Input: Query $\boldsymbol{x}_t$ and parameters Φ of the neural network.

Output: Hash code $\boldsymbol{b}_t$ of $\boldsymbol{x}_t$.

1. Calculate the network output by the forward propagation algorithm;
2. Calculate the Hash code of $\boldsymbol{x}_t$ by Equation (9).

3 Experiment and Analysis

This section verifies the effectiveness of the proposed method, including Mean Average Precision (MAP), parameter sensitivity analysis and ablation study. In this paper, PyTorch is used to build a deep Hashing model, and the model parameters are optimized by stochastic gradient descent with momentum method, in which the batch size, momentum parameter and learning rate are set as 16, 0.9 and 0.001, respectively. For fair comparison with other Hashing models, this method directly cuts the original image to a size of 224 × 224 as the input of the model, without any data augmentation, and the feature vector is extracted from fc-7 of VGG-F. Then, the feature vector is input into the Hashing layer to obtain the Hash code of each original image, where the Hashing layer is the last fully connected layer of the model.

In two benchmark datasets NUSWIDE and FLICKR25K, our method is compared with some effective deep learning methods and traditional shallow-layer methods. The common evaluation criterion MAP is employed to measure the performance of this method.

MAP is the mean of Average Precision (*AP*) for each query:

$$AP = \frac{1}{N}\sum_{r=1}^{R} P(r)\delta(r),$$

where N is the number of samples related to the query label, $P(r)$ is the retrieval precision of the first r samples, and $\delta(r)$ is the relevance of the rth sample to the query. R is set as 5 000 in this paper and we determine that if at least one label of two samples is the same, then the two samples are related.

3.1 Datasets and experiment environment

This experiment runs on a server equipped with an operating system of Linux version 4.4.0-1164GB-generic (buildd@lgw01-amd64-021) (gcc version 5.4.0 20160609 (Ubuntu 5.4.0-6 Ubuntu1~16.04.9)), a processor of Intel(R) Xeon(R) Silver 4210CPU@2.20GHz and a memory of 64 GB.

FLICKR25K contains 25 000 images collected from Flickr website, which are divided into 24 categories. A total of 2 000 images are randomly selected as the test set, and the rest as the retrieval set, from which 10 000 images are randomly selected as the training set.

NUSWIDE covers 269 648 images belonging to 81 categories. The data subset used in this paper contains 10 most common labels. A total of 5 000 images are randomly selected as the test set, and the rest images as the retrieval set, from which 5 000 images are randomly selected as the training set.

3.2 Results in FLICKR25K

Table 2 presents the MAP of our method and other methods on FLICKR25K when the length of Hash codes changes from 16 bits to 128 bits. Our method outperforms other methods. In the case of 16-bit, 32-bit, 64-bit and 128-bit Hash codes, the MAP of our method is 5.17%, 6.46%, 6.70% and 7.45% higher than that of the second-best SSDH, respectively. Among other Hashing methods, ITQ[11], Spectral Hashing (SH)[29], Density Sensitive Hashing (DSH)[30], Spherical Hashing (SpH)[31], and Scalable Graph Hashing (SGH)[21] are traditional shallow-layer methods, while DeepBit[27] and SSDH[14] are based on deep models. By comparison, we find that some non-deep Hashing methods have higher MAP than deep ones. This may be because the deep Hashing method cannot fully utilize the feature expression ability of the deep network in the case of lacking supervised information, and it is prone to over-fitting to the local minimum, thus showing poor performance. The self-supervised Hashing based on contrastive learning and with the regularization of Hash codes achieves the best MAP results.

Table 2 MAP of different code lengths on FLICKR25K

Algorithm	FLICKR25K (bits)			
	16	32	64	128
ITQ	0.6492	0.6518	0.6546	0.6577
SH	0.6091	0.6105	0.6033	0.6014
DSH	0.6452	0.6547	0.6551	0.6557
SpH	0.6119	0.6315	0.6381	0.6451
SGH	0.6362	0.6283	0.6253	0.6206
DeepBit	0.5934	0.5933	0.6199	0.6349
SSDH	0.7240	0.7276	0.7377	0.7343
Ours	**0.7757**	**0.7922**	**0.8047**	**0.8088**

3.3 Results in NUSWIDE

Table 3 exhibits the MAP of our method and other methods on NUSWIDE when the length of Hash codes changes from 16 bits to 128 bits.

Table 3 MAP of different code lengths on NUSWIDE

Algorithm	NUSWIDE (bits)			
	16	32	64	128
ITQ	0.5270	0.5241	0.533 4	0.5398
SH	0.4350	0.4129	0.406 2	0.4100
DSH	0.5123	0.5118	0.511 0	0.5267
SpH	0.4458	0.4537	0.492 6	0.5000
SGH	0.4994	0.4869	0.485 1	0.4945
DeepBit	0.3844	0.4341	0.446 1	0.4917
SSDH	0.6374	0.6768	0.682 9	0.6831
Ours	**0.7070**	**0.7397**	**0.761 3**	**0.7868**

Table 3 reveals that the method proposed in this paper performs better than other methods. The MAP of our method is 6.96%, 6.29%, 7.84% and 10.37% higher than that of SSDH in the case of 16-bit, 32-bit, 64-bit and 128-bit Hash codes, respectively. Longer Hash codes can process more information and thus have a higher MAP. In addition, the difficulty of searching in NUSWIDE surges since its size is about 10 times that of FLICKR25K. Therefore, the MAP in NUSWIDE is smaller than that in FLICKR25K at the same length of Hash codes.

3.4 Ablation studies

In this section, ablation studies are conducted to verify the effectiveness of each part of the proposed algorithm. The method is divided into three components (Table 4), which are whether to add local semantic structure information, whether to add contrastive learning loss and whether to add regular term loss. Different components are combined to yield the ablation study results and reflect the influence of each component on the results.

Table 4 Three main components in ablation studies

Symbol	Meaning
C1	Add local semantic structure information
C2	Add contrastive learning loss
C3	Add regular term loss

The results of ablation studies on FLICKR25K and with 16-bit and 32-bit Hash codes are respectively demonstrated in Tables 5 and 6.

Table 5 MAP of our method's variants on FLICKR25K at 16-bit codes

Method	C1	C2	C3	MAP
Our method	√			0.7435
	√	√		0.7659
	√	√	√	0.7757

Table 6 MAP of our method's variants on FLICKR25K at 32-bit codes

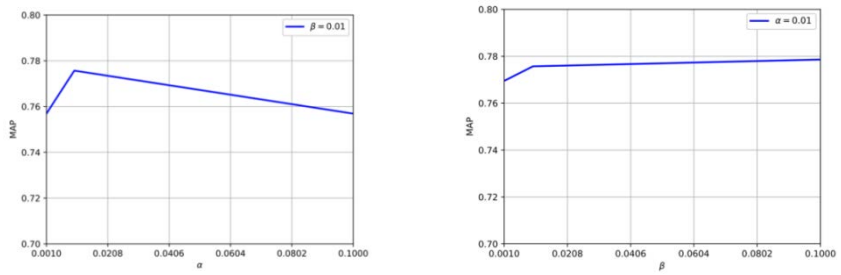
Method	C1	C2	C3	MAP
Our method	√			0.7517
	√	√		0.7890
	√	√	√	0.7922

Tables 5 and 6 demonstrate that the precision of our method can be improved after regular term and contrastive learning loss term are added. Besides semantic structure information, contrastive learning loss is added to the model. By applying a momentum contrastive learning algorithm to Hash codes, the model can learn more accurate Hash codes. As such, the precision of the model is significantly improved. In addition to semantic structure information and contrastive learning loss, regular term loss is also added. The discrimination of Hash codes can be enhanced if the Hash codes approach 1 or −1 as much as possible. Therefore, keeping the semantic structure information of data and improving the discrimination of Hash code in the Hamming space

facilitate the performance improvement, which verifies the effectiveness of the method proposed in this paper.

3.5 Parameter sensitivity analysis

In this section, the sensitivity analysis of the main hyperparameters α, β and τ in the proposed method is carried out. The experiment is carried out with 16-bit Hash codes on FLICKR25K. Figure 3(a) and 3(b) respectively correspond to the cases with β fixed at 0.01 and α varying from 0.001 to 0.1, and α fixed at 0.01 and β ranging from 0.001 to 0.1.



(a) Variation of MAP with α (b) Variation of MAP with β

Figure 3 Loss hyperparameters of 16-bit codes in FLICKR25K

Figure 3 illustrates that in the case of a fixed β, MAP first increases and then decreases with the rise in α, and it reaches the maximum when α is 0.01. In the case of α fixed at 0.01, MAP climbs up and then declines with the increase in β, and it peaks when β is 0.1. Comparison of the influences of α and β on MAP indicates that α has a greater impact on the experimental results, while β can improve the experimental results to some extent.

To more clearly analyze the impact of α and β, we plot a 3D figure which α ranging from 0.001 to 0.1, β ranging from 0.001 to 0.1. In Figure 4, we can easily find the best hyperparameter combination, which $\alpha = 0.01$, $\beta = 0.01$. In other experiments, this paper fixed α=0.01 and β=0.01.

With $\alpha = 0.01$, $\beta = 0.01$ and τ varying within 0–0.5, the results are displayed in Figure 5.

τ is a temperature parameter that controls the concentration of data distribution. Figure 5 reveals that with fixed α and β, MAP grows before dropping with the increase in τ, and it peaks when τ is 0.07. In other experiment, $\alpha = 0.01$, $\beta = 0.01$ and $\tau = 0.07$.

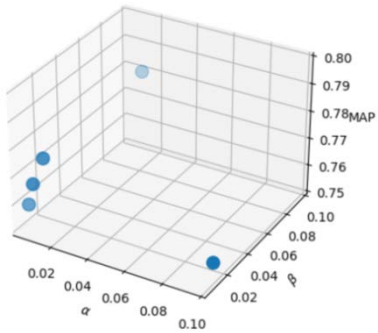


Figure 4 3D plot of MAP with respect to hyperparameters α and β

3.6 Time complexity analysis

Time complexity is the number of operations of the model, which can be measured by FLOating-Point operation (FLOP).

The time complexity of all convolutional layers is

$$O\left(\sum_{l=1}^{d} n_{l-1} \cdot s_l^2 \cdot n_l \cdot m_l^2\right)$$

where l is the subscript of a convolutional layer. d is the number of the convolutional layers. n_l is the number of convolution kernels in the lth network. n_{l-1} is the number of input channels in the lth network. s_l is the size of a convolution kernel. m_l is the size of the output feature map. The training time of each image is about three times as long as the test time (one time in forward propagation and two times in back propagation). The total time complexity of VGG-F and Hash code layer in the training is calculated to be 31.0 GFlops, where the time consumption of the fully connected and pooling layers accounts for 0.8%.

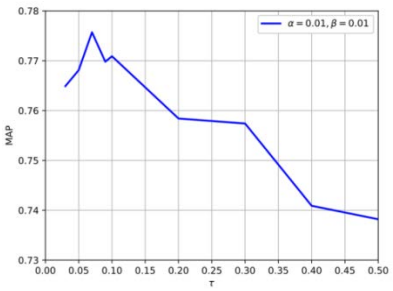


Figure 5 Temperature hyperparameter of 16-bit codes in FLICKR25K

4 Conclusion and Prospects

In view of the problems in unsupervised Hashing, this paper proposed a novel deep unsupervised Hashing method based on semantic structure preserving and instanced discrimination. To improve the discriminability of Hash codes, we adopted self-supervised learning to capture the semantic structure and guided the learning to Hash. Experiments were carried out in two datasets commonly used to evaluate Hashing methods, and extensive experimental design and analyses verified the effectiveness of the proposed method. The future research will focus on the design of more effective self-supervised learning tasks and relevant loss functions as well as the optimization of the algorithm to increase the training speed.

References

[1] Wu GS, Lin ZJ, Ha JG, Liu L, Ding GG, Zhang BC, Shen JL. Unsupervised deep Hashing via binary latent factor models for large- scale cross-modal retrieval. Proc. of the 27th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2018). 2018. 2854–2860.

[2] Aytekin AM, Aytekin T. Real-time recommendation with locality sensitive Hashing. Journal of Intelligent Information Systems, 2019, 53(1): 1–26.

[3] Sun JD, Wang J, Yuan H, Liu XC, Liu J. Unequally weighted video Hashing for copy detection. In: Li SP, ed. Proc. of the 19th Int'l Conf. on Multimedia Modeling, Advances in Multimedia Modeling (MMM 2013). Part I. Huangshan, Berlin, Heidelberg: Springer-Verlag, 2013. 546–557.

[4] Gionis A, Indyk P, Motwani R. Similarity search in high dimensions via Hashing. In: Atkinson MP, ed. Proc. of the 25th Int'l Conf. on Very Large Data Bases (VLDB'99). Edinburgh: Morgan Kaufmann publishers, 1999, 99(6): 518–529.

[5] Wang JD, Zhang T, Song JK, Sebe N, Shen HT. A survey on learning to Hash. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2017, 40(4): 769–790.

[6] Lin GS, Shen CH, Shi QF, van den Hengel A, Suter D. Fast supervised Hashing with decision trees for high-dimensional data. In: Proc. of the 2014 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2014). Columbus: IEEE Computer Society, 2014. 1963–1970.

[7] Liu W, Wang J, Ji RR, Jiang YG, Chang SF. Supervised Hashing with kernels. Proc. of the 2012 IEEE Conf. on Computer Vision and Pattern Recognition. Providence: IEEE Computer Society, 2012. 2074–2081.

[8] Shen FM, Zhou X, Yang Y, Song JK, Shen HT, Tao DC. A fast optimization method for general binary code learning. IEEE Trans. on Image Processing, 2016, 25(12): 5610–5621.

[9] Wang J, Kumar S, Chang SF. Semi-Supervised Hashing for large-scale search. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2012, 34(12): 2393–2406.

[10] Mu Y, Shen J, Yan S. Weakly-supervised Hashing in kernel space. Proc. of the 23rd IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2010). San Francisco: IEEE Computer Society, 2010. 3344–3351.

[11] Gong YC, Lazebnik S, Gordo A, Perronnin F. Iterative quantization: A procrustean approach to learning binary codes for large- scale image retrieval. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2012, 35(12): 2916–2929.

[12] Hu MQ, Yang Y, Shen FM, Xie N, Shen HT. Hashing with angular reconstructive embeddings. IEEE Trans. on Image Processing, 2017, 27(2): 545–555.

[13] Liu W, Mu C, Kumar S, Chang SF. Discrete graph Hashing. In: Ghahramani Z, ed. Proc. of the Advances in Neural Information Processing Systems 27: Annual Conf. on Neural Information Processing Systems 2014. 2014. 3419–3427.

[14] Yang EK, Deng C, Liu TL, Liu W, Tao DC. Semantic structure-based unsupervised deep Hashing. Proc. of the 27th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2018). 2018. 1064–1070.

[15] Kumar S, Udupa R. Learning Hash functions for cross-view similarity search. In: Walsh T, ed. Proc. of the 22nd Int'l Joint Conf. on Artificial Intelligence (IJCAI 2011). Barcelona: IJCAI/AAAI, 2011. 1360–1365.

[16] Song JK, Yang Y, Yang Y, Huang Z, Shen HT. Inter-Media Hashing for large-scale retrieval from heterogeneous data sources. In: Ross KA, ed. Proc. of the ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2013). New York: ACM, 2013. 785–796.

[17] Zhu XF, Huang Z, Shen HT, Zhao X. Linear cross-modal Hashing for efficient multimedia search. In: Jaimes A, ed. Proc. of the ACM Multimedia Conf. (MM 2013). Barcelona: ACM, 2013. 143–152.

[18] Shen FM, Shen CH, Liu W, Shen HT. Supervised discrete Hashing. Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2015). Boston: IEEE Computer Society, 2015. 37–45.

[19] Wang J, Liu W, Sun AX, Jiang YG. Learning Hash codes with listwise supervision. Proc. of the IEEE Int'l Conf. on Computer Vision (ICCV 2013). Sydney: IEEE Computer Society, 2013. 3032–3039.

[20] Xia RK, Pan Y, Lai HJ, Liu C, Yan SC. Supervised Hashing for image retrieval via image representation learning. In: Brodley CE, ed. Proc. of the 28th AAAI Conf. on Artificial Intelligence. Quebec City: AAAI, 2014. 2156–2162.

[21] Jiang QY, Li WJ. Scalable graph Hashing with feature transformation. In: Yang Q, ed. Proc. of the 24th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2015). Buenos Aires: AAAI, 2015. 2248–2254.

[22] Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2013,35(8):1798–1828.

[23] He KM, Fan HQ, Wu YX, Xie SN, Girshick RB. Momentum contrast for unsupervised visual representation learning. Proc. of the 2020 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR 2020). Seattle: IEEE, 2020. 9729–9738.

[24] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[25] Salakhutdinov R, Hinton G. Semantic Hashing. Int'l Journal of Approximate Reasoning, 2009, 50(7): 969–978.

[26] Krizhevsky A, Hinton GE. Using very deep autoencoders for content-based image retrieval. Proc. of the 19th European Symp. on Artificial Neural Networks (ESANN 2011). 2011.

[27] Lin K, Lu JW, Chen CS, Zhou J. Learning compact binary descriptors with unsupervised deep neural networks. Proc. of the 2016 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2016). Las Vegas: IEEE Computer Society, 2016. 1183–1192.

[28] Li C, Deng C, Li N, Liu W, Gao XB, Tao DC. Self-Supervised adversarial Hashing networks for cross-modal retrieval. Proc. of the 2018 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2018). Salt Lake City: IEEE Computer Society, 2018. 4242–4251.

[29] Weiss Y, Torralba A, Fergus R. Spectral Hashing. In: Koller D, ed. Proc. of the Advances in Neural Information Processing Systems 21, 22nd Annual Conf. on Neural Information Processing Systems. Vancouver: Curran Associates, Inc., 2008. 1753–1760.

[30] Jin ZM, Li C, Lin Y, Cai D. Density sensitive Hashing. IEEE Trans. on Cybernetics, 2013, 44(8): 1362–1371.

[31] Heo JP, Lee YW, He JF, Chang SF, Yoon SE. Spherical Hashing. Proc. of the 2012 IEEE Conf. on Computer Vision and Pattern Recognition. Providence: IEEE Computer Society, 2012.

Changsheng Li, Ph.D., professor, doctoral supervisor. His research field is machine learning.

Ye Yuan, Ph.D., professor, doctoral supervisor, senior member of CCF. His research field is database.

Qixing Min, bachelor. Her research field is video behavior analysis.

Guoren Wang, Ph.D., professor, doctoral supervisor, distinguished member of CCF. His research field is database.

Yurong Cheng, Ph.D., associate professor, doctoral supervisor, professional member of CCF. Her research field is database.