Deep Discrete Hashing with Self-supervised Pairwise Labels

Jingkuan Song, Tao He, Hangbo Fan, and Lianli Gao(✉)

University of Electronic Science and Technology of China,
2006th Xiyuan Ave, Chengdu, China
jingkuan.song@gmail.com, tao.he@gmail.com, hbfan@gmail.com,
lianli.gao@uestc.edu.cn

Abstract. Hashing methods have been widely used for applications of large-scale image retrieval and classification. Non-deep hashing methods using handcrafted features have been significantly outperformed by deep hashing methods due to their better feature representation and end-to-end learning framework. However, the most striking successes in deep hashing have mostly involved discriminative models, which require labels. In this paper, we propose a novel unsupervised deep hashing method, named Deep Discrete Hashing (DDH), for large-scale image retrieval and classification. In the proposed framework, we address two main problems: (1) how to directly learn discrete binary codes? (2) how to equip the binary representation with the ability of accurate image retrieval and classification in an unsupervised way? We resolve these problems by introducing an intermediate variable and a loss function steering the learning process, which is based on the neighborhood structure in the original space. Experimental results on standard datasets (CIFAR-10, NUS-WIDE, and Oxford-17) demonstrate that our DDH significantly outperforms existing hashing methods by large margin in terms of mAP for image retrieval and object recognition. Code is available at https://github.com/htconquer/ddh.

1 Introduction

Due to the popularity of capturing devices and the high speed of network transformation, we are witnessing the explosive growth of images, which attracts great attention in computer vision to facilitate the development of multimedia search [35,42], object segmentation [22,34], object detection [26], image understanding [4,33] etc. Without a doubt, the ever growing abundance of images brings an urgent need for more advanced large-scale image retrieval technologies. To date, high-dimensional real-valued features descriptors (e.g., deep Convolutional Neural Networks (CNN) [30,37,39] and SIFT descriptors) demonstrate superior discriminability, and bridge the gap between low-level pixels and high-level semantic information. But they are less efficient for large-scale retrieval due to their high dimensionality.

Therefore, it is necessary to transform these high-dimensional features into compact binary codes which enable machines to run retrieval in real-time and

M. Ceci et al. (Eds.): ECML PKDD 2017, Part I, LNAI 10534, pp. 223–238, 2017.
https://doi.org/10.1007/978-3-319-71249-9_14

with low memory. Existing hashing methods can be classified into two categories: *data-independent* and *data-dependent*. For the first category, hash codes are generated by randomly projecting samples into a feature space and then performing binarization, which is independent of any training samples. On the other hand, *data-dependent* hashing methods learn hash functions by exploring the distribution of the training data and therefore, they are also called learning to hashing methods (L2H) [43]. A lot of L2H methods have been proposed, such as Spectral hashing (SpeH) [45], iterative quantization (ITQ) [9], Multiple Feature Hashing (MFH) [36], Quantization-based Hashing (QBH) [32], K-means Hashing (KMH) [12], DH [24], DPSH [20], DeepBit [23], etc. Actually, those methods can be further divided into two categories: supervised methods and unsupervised methods. The difference between them is whether to use supervision information, e.g., classification labels. Some representative unsupervised methods include ITQ, Isotropic hashing [16], and DeepBit which achieves promising results, but are usually outperformed by supervised methods. By contrast, the supervised methods take full advantage of the supervision information. One representative is DPSH [20], which is the first method that can perform simultaneous feature learning and hash codes learning with pairwise labels. However, the information that can be used for supervision is also typically scarce.

To date, hand-craft floating-point descriptors such as SIFT, Speeded-up Robust Features (SURF) [2], DAISY [41], Multisupport Region Order-Based Gradient Histogram (MROGH) [8], the Multisupport Region Rotation and Intensity Monotonic Invariant Descriptor (MRRID) [8] etc., are widely utilized to support image retrieval since they are distinctive and invariant to a range of common image transformations. In [29], they propose a content similarity based fast reference frame selection algorithm for reducing the computational complexity of the multiple reference frames based inter-frame prediction. In [40], they develop a so-called correlation component manifold space learning (CCMSL) to learn a common feature space by capturing the correlations between the heterogeneous databases. Many attempts [21,25] were focusing on compacting such high quality floating-point descriptors for reducing computation time and memory usage as well as improving the matching efficiency. In those methods, the floating-point descriptor construction procedure is independent of the hash codes learning and still costs a massive amounts of time-consuming computation. Moreover, such hand-crafted feature may not be optimally compatible with hash codes learning. Therefore, these existing approaches might not achieve satisfactory performance in practice.

To overcome the limitation of existing hand-crafted feature based methods, some deep feature learning based deep hashing methods [7,10,11,20,46,47] have recently been proposed to perform simultaneous feature learning and hash-code learning with deep neural networks, which have shown better performance than traditional hashing methods with hand-crafted features. Most of these deep hashing methods are supervised whose supervision information is given as triplet or pairwise labels. An example is the deep supervised hashing method by Li *et al.* [20], which can simultaneously learn features and hash codes. Another example

is Supervised Recurrent Hashing (SRH) [10] for generating hash codes of videos. Cao *et al.* [3] proposed a continuous method to learn binary codes, which can avoid the relaxation of binary constraints [10] by first learning continuous representations and then thresholding them to get the hash codes. They also added weight to data for balancing similar and dissimilar pairs.

In the real world, however, the vast majority of training data do not have labels, especially for scalable dataset. To the best of our knowledge, DeepBit [23] is the first to propose a deep neural network to learn binary descriptors in an unsupervised manner, by enforcing three criteria on binary codes. It achieves the state-of-art performance for image retrieval, but DeepBit does not consider the data distribution in the original image space. Therefore, DeepBit misses a lot of useful unsupervised information.

So can we obtain the pairwise information by exploring the data distribution, and then use this information to guide the learning of hash codes? Motivated by this, in this paper, we propose a Deep Discrete Hashing (DDH) with pseudo pairwise labels which makes use the self-generated labels of the training data as supervision information to improve the effectiveness of the hash codes. It is worth highlighting the following contributions:

1. We propose a general end-to-end learning framework to learn discrete hashing code in an unsupervised way to improve the effectiveness of hashing methods. The discrete binary codes are directly optimized from the training data. We solve the discrete hash, which is hard to optimize, by introducing an intermediate variable.
2. To explore the data distribution of the training images, we learn on the training dataset and generate the pairwise information. We then train our model by using this pairwise information in a self-supervised way.
3. Experiments on real datasets show that DDH achieves significantly better performance than the state-of-the-art unsupervised deep hashing methods in image retrieval applications and object recognition.

2 Our Method

Given N images, $\mathbf{I} = \{\mathbf{I}_i\}_{i=1}^{N}$ without labels, our goal is to learn their compact binary codes $\mathbf{B}$ such that: (a) the binary codes can preserve the data distribution in the original space, and (b) the discrete binary codes could be computed directly.

As shown in Fig. 1, our DDH consists of two key components: construction of pairwise labels, and hashing loss definition. For training, we first construct the neighborhood structure of images and then train the network. For testing, we obtain the binary codes of an image by taking it as an input. In the remainder of this section, we first describe the process of constructing the neighborhood structure, and then introduce our loss function and the process of learning the parameters.

2.1 Construction of Pairwise Labels

In our unsupervised approach, we propose to exploit the neighborhood structure of the images in a feature space as information source steering the process of hash learning. Specifically, we propose a method based on the K-Nearest Neighbor (KNN) concept to create a neighborhood matrix $\mathbf{S}$. Based on [13], we extract 2,048-dimensional features from the pool5-layer, which is last layer of ResNet [13]. This results in the set $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$ where $\mathbf{x}_i$ is the feature vector of image $\mathbf{I}_i$.

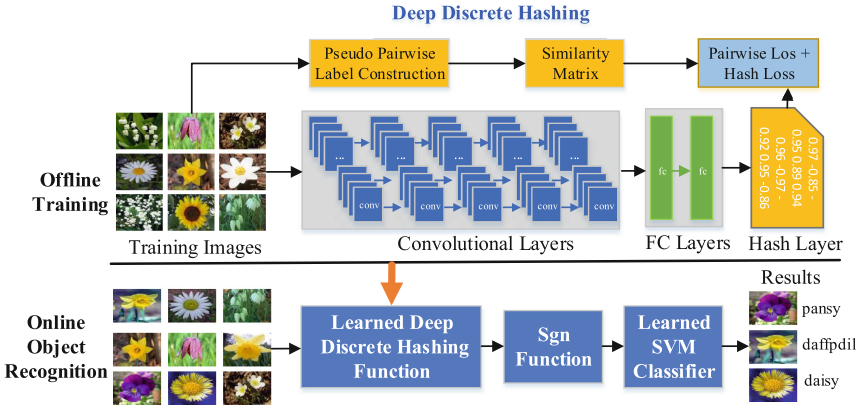


Fig. 1. The structure of our end-to-end framework. It has two components, construction of pairwise labels, and hashing loss definition. We first construct the neighborhood structure of images and then train the network based on the define loss function. We utilize the deep neural network to extract the features of the images.

For the representation of the neighboring structure, our task is to construct a matrix $\boldsymbol{S} = (s_{ij})_{i,j=1}^{N}$, whose elements indicate the similarity ($s_{ij} = 1$) or dissimilarity ($s_{ij} = -1$) of any two images i and j in terms of their features $\mathbf{x}_i$ and $\mathbf{x}_j$.

We compare images using cosine similarity of the feature vectors. For each image, we select K_1 images with the highest cosine similarity as its neighbors. Then we can construct an initial similarity matrix $\mathbf{S}_1$:

$$(S_1)_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_j \text{ is } K_1\text{-NN of } \mathbf{x}_i \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Here we use $\mathbf{L}_1, \mathbf{L}_2, \ldots, \mathbf{L}_N$ to denote the ranking lists of points $\mathbf{I}_1$,$\mathbf{I}_2$,...,$\mathbf{I}_N$ by K_1-NN. The precision-recall curve in Fig. 2 indicates the quality of the constructed neighborhood for different values of K_1. Due to the rapidly decreasing precision with the increase of K_1, creating a large-enough neighborhood by simply increasing K_1 is not the best option. In order to find a better approach, we

borrow the ideas from the domain of graph modeling. In an undirected graph, if a node v is connected to a node u and if u is connected to a node w, we can infer that v is also connected to w. Inspired by this, if we treat every training image as a node in an undirected graph, we can expand the neighborhood of an image i by exploring the neighbors of its neighbors. Specifically, if $\mathbf{x}_i$ connects to $\mathbf{x}_j$ and $\mathbf{x}_j$ connects to $\mathbf{x}_k$, we can infer that $\mathbf{x}_i$ has the potential to be connected to $\mathbf{x}_k$ as well.

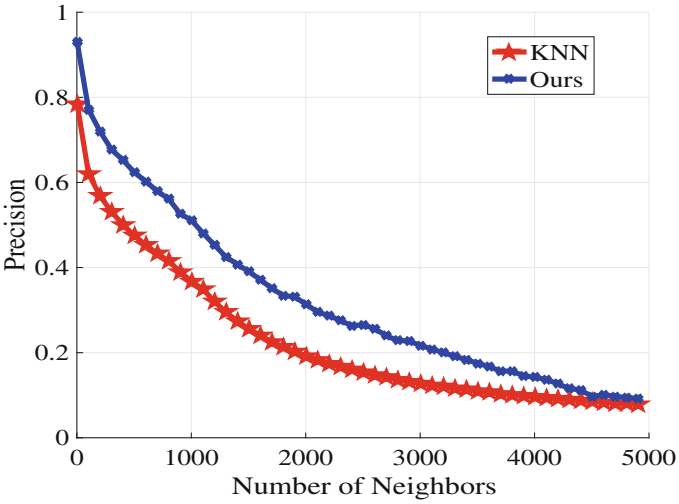


Fig. 2. Precision of constructed labels on cifar-10 dataset with different K, and different methods.

Based on the above observations, we construct $\mathbf{S}_1$ using the deep CNN features. If we only use the constructed labels by $\mathbf{S}_1$, each image has too few positive labels with high precision. So we increase the number of neighbors based on $\mathbf{S}_1$ to obtain more positive labels. Specifically, we calculate the similarity of two images by comparing the two ranking lists of K_1-NN using the expression $\frac{1}{||\mathbf{L}_i - \mathbf{L}_j||^2}$. Actually, if two images have the same labels, they should have a lot of intersection points based on K_1-NN, i.e., they have similar K_1-NN ranking list. Then we again construct a ranking list of K_2 neighbors, based on which we generate the second similarity matrix $\mathbf{S}_2$ as:

$$(\mathbf{S}_2)_{ij} = \begin{cases} 1, & \text{if } \mathbf{L}_j \text{ is } K_2\text{-NN of } \mathbf{L}_i \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Finally, we construct the neighborhood structure by combining the direct and indirect similarities from the two matrices together. This results in the final similarity matrix $\mathbf{S}$:

$$S_{ij} = \begin{cases} 1, & \text{if } (\mathbf{S}_2)_{ik} = 1 \textit{ and } j \textit{ in } \mathbf{L}_k \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where the $\mathbf{L}_k$ is the ranking list after K_1-NN. The whole algorithm is shown in Algorithm 1. After the two steps KNN, the constructed label precision is shown in Fig. 2. We note that we could have also omitted this preprocessing step and construct the neighborhood structure directly during the learning of our neural network. We found, however, that the construction of neighborhood structure is time-consuming, and that updating of this structure based on the updating of image features in each epoch does not have significant impact on the performance. Therefore, we chose to obtain this neighborhood structure as described above and fix it for the rest of the process.

Algorithm 1. Construction of neighborhood structure

Input: Images $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, the number of neighbors K_1, the number of neighbors K_2 for the neighbors expansion;
Output: Neighborhood matrix $\mathbf{S} = \{s_{ij}\}$;
First ranking: Use cosine similarity to generate the index of K_1-NN of each image $L_1, L_2, \ldots, L_N$;
Neighborhood expansion:
for i=1,...,N **do**
  Initialize $num \leftarrow \emptyset$;
  for $j = 1, \ldots N$ **do**
    $num_j \leftarrow$ the size of $L_i \cap L_j$;
  end for
  Sort num by descending order and keep the top K_2 $\{L_j\}$;
  Set new $L'_i \leftarrow$ union of the top K_2 $\{L_j\}$;
end for
for j=1,...,N **do**
  Construct $\mathbf{S}$ with new L'_j base on Eq. 3;
end for
return $\mathbf{S}$;

2.2 Architecture of Our Network

We introduce an unsupervised deep framework, dubbed Deep Discrete Hashing (DDH), to learn compact yet discriminative binary descriptors. The framework includes two main modules, feature learning part and hash codes learning part, as shown in Fig. 1. More specifically, for the feature learning, we use a similar network as in [48], which has seven layers and the details are shown in Table 1. In the experiment, we can easily replace the CNN-F network with other deep networks such as [13,18,38]. Our framework has two branches with the shared weights and both of them have the same weights and same network structure.

We discard the last softmax layer and replace it with a hashing layer, which consists of a fully connected layer and a sgn activation layer to generate compact codes. Specifically, the output of the $full_7$ is firstly mapped to a L-dimensional real-value code, and then a binary hash code is learned directly, by converting

the L-dimensional representation to a binary hash code $\mathbf{b}$ taking values of either $+1$ or -1. This binarization process can only be performed by taking the sign function $\mathbf{b} = sgn(\mathbf{u})$ as the activation function on top of the hash layer.

$$\mathbf{b} = \text{sgn}(\mathbf{u}) = \begin{cases} +1, \ if \ \mathbf{u} \geq 0 \\ -1, \quad \text{otherwise} \end{cases} \tag{4}$$

Table 1. The configuration of our framework

Layer	Configure
$conv_1$	filter $64 \times 11 \times 11$, stride 4×4, pad 0, LRN, pool 2×2
$conv_2$	filter $256 \times 5 \times 5$, stride 1×1, pad 2, LRN, pool 2×2
$conv_3$	filter $256 \times 3 \times 3$, stride 1×1, pad 1
$conv_4$	filter $256 \times 3 \times 3$, stride 1×1, pad 1
$conv_5$	filter $256 \times 3 \times 3$, stride 1×1, pad 1, pool 2×2
$full_6$	4096
$full_7$	4096
hash layer	L

2.3 Objective Function

Suppose we denote the binary codes as $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^{N}$ for all the images. The neighborhood structure loss models the loss in the similarity structure in data, as revealed in the set of neighbors obtained for an image by applying the hash code of that image. We define the loss function as below:

$$\min J_1 = \frac{1}{2} \sum_{s_{ij} \in S} \left(\frac{1}{L} \mathbf{b}_i{}^T \mathbf{b}_j - s_{ij} \right)^2 \tag{5}$$

where L is the length of hashing code and $s_{ij} \in \{-1, 1\}$ indicates the similarity of image i and j. The goal of optimizing for this loss function is clearly to bring the binary codes of similar images as close to each other as possible.

We also want to minimize the quantization loss between the learned binary vectors $\mathbf{B}$ and the original real-valued vectors $\mathbf{Z}$. It is defined as:

$$\min J_2 = \frac{1}{2} \sum_{i=1}^{N} \|\mathbf{z}_i - \mathbf{b}_i\|^2 \tag{6}$$

where $\mathbf{z}_i$ and $\mathbf{b}_i$ are the real-valued representation and binary codes of the i-th image in the hashing layer. Then we can obtain our final objective function as:

$$\min J = J_1 + \lambda_1 J_2, \qquad \mathbf{b}_i \in \{-1, 1\}^L, \quad \forall i = 1, 2, 3, \ldots N \tag{7}$$

where λ_1 is the parameter to balance these two terms.

Obviously, the problem in (7) is a discrete optimization problem, which is hard to solve. LFH [48] solves it by directly relaxing $\mathbf{b}_i$ from discrete to continuous, which might not achieve satisfactory performance [15]. In this paper, we design a novel strategy which can solve the problem 5 by introducing an intermediate variable. First, we reformulate the problem in 5 as the following equivalent one:

$$\begin{aligned} \min J &= \frac{1}{2}\sum_{s_{ij}\in S}\left(\frac{1}{L}{\mathbf{b}_i}^T\mathbf{b}_j - s_{ij}\right)^2 + \frac{\lambda_1}{2}\sum_{i=1}^{N}\|\mathbf{z}_i - \mathbf{b}_i\|^2 \\ s.t \quad & \mathbf{u}_i = \mathbf{b}_i, \forall i = 1, 2, 3, \ldots N \\ & \mathbf{u}_i \in \mathbb{R}^{L\times 1}, \forall i = 1, 2, 3, \ldots N \\ & \mathbf{b}_i \in \{-1, 1\}^L, \forall i = 1, 2, 3, \ldots N \end{aligned} \tag{8}$$

where $\mathbf{u}_i$ is an intermediate variable and $\mathbf{b}_i = sgn(\mathbf{u}_i)$. To optimize the problem in 8, we can optimize the following regularized problem by moving the equality constraints in 8 to the regularization terms:

$$\begin{aligned} \min J &= \frac{1}{2}\sum_{s_{ij}\in \mathbf{S}}\left(\frac{1}{L}{\mathbf{u}_i}^T\mathbf{u}_j - s_{ij}\right)^2 + \frac{\lambda_1}{2}\sum_{i=1}^{N}\|\mathbf{z}_i - \mathbf{b}_i\|^2 + \frac{\lambda_2}{2}\sum_{i=1}^{N}\|\mathbf{b}_i - \mathbf{u}_i\|^2 \\ s.t \quad & \mathbf{u}_i \in \mathbb{R}^{L\times 1}, \forall i = 1, 2, 3, \ldots N \\ & \mathbf{b}_i \in \{-1, 1\}^L, \forall i = 1, 2, 3, \ldots N \end{aligned} \tag{9}$$

where λ_2 is the hyper-parameter for the regularization term. Actually, introducing an intermediate variable $\mathbf{u}$ is equivalent to adding another full-connected layer between $\mathbf{z}$ and $\mathbf{b}$ in the hashing layer. To reduce the complexity of our model, we let $\mathbf{z} = \mathbf{u}$, and then we can have a simplified objective function as:

$$\begin{aligned} \min J &= \frac{1}{2}\sum_{s_{ij}\in \mathbf{S}}\left(\frac{1}{K}{\mathbf{z}_i}^T\mathbf{z}_j - s_{ij}\right)^2 + \frac{\lambda_1}{2}\sum_{i=1}^{N}\|\mathbf{z}_i - \mathbf{b}_i\|^2 \\ s.t \quad & \mathbf{b}_i \in \{-1, 1\}^L, \forall i = 1, 2, 3, \ldots N \end{aligned} \tag{10}$$

Equation 10 is not discrete and $\mathbf{z}_i$ is derivable, so we can use back-propagation (BP) to optimize it.

2.4 Learning

To learning DDH Model, we need to obtain the parameters of neural networks. We set

$$\begin{aligned} \mathbf{z}_i &= \mathbf{W}^T\phi\left(x_i;\theta\right) + \mathbf{c} \\ \mathbf{b}_i &= sgn(\mathbf{z}_i) = sgn(\mathbf{W}^T\phi\left(x_i;\theta\right) + \mathbf{c}) \end{aligned} \tag{11}$$

where θ denotes all the parameters of CNN-F network for learning the features. $\phi\left(x_i;\theta\right)$ denotes the output of the $full_7$ layer associated with image x_i. $\mathbf{W} \in$

$\mathbb{R}^{4096\times L}$ denotes hash layer weights matrix, and $\mathbf{C}\in\mathbb{R}^{L\times 1}$ is a bias vector. We add regularization terms on the parameters and change the loss function with 10 constrains as:

$$\begin{aligned}\min J &= \tfrac{1}{2}\sum_{s_{ij}\in S}\left(\tfrac{1}{L}\Theta_{ij}-s_{ij}\right)^2\\ &+\tfrac{\lambda_1}{2}\sum_{i=1}^{N}\left\|\mathbf{b}_i-\left(\mathbf{W}^T\phi\left(x_i;\theta\right)+\mathbf{c}\right)\right\|^2\\ &+\tfrac{\lambda_2}{2}(\|\mathbf{W}\|_F^2+\|\mathbf{c}\|_F^2)^2\end{aligned} \tag{12}$$

where $\Theta_{ij}={z_i}^T z_j$, λ_1 and λ_2 are two parameters to balance the effect of different terms. Stochastic gradient descent (SGD) is used to learn the parameters. We use CNN-F network trained on ImageNet to initialize our network. In particular, in each iteration we sample a mini-batch of points from the whole training set and use back-propagation (BP) to optimize the whole network. Here, we compute the derivatives of the loss function as follows:

$$\frac{\partial J}{\partial \mathbf{z}_i}=\frac{1}{L^2}({\mathbf{z}_i}^T\mathbf{z}_j)\mathbf{z}_j-\frac{1}{L}s_{ij}\mathbf{z}_j+\lambda_1(\mathbf{z}_i-\mathbf{b}_i) \tag{13}$$

2.5 Out-of-Sample Extension

After the network has been trained, we still need to obtain the hashing codes of the images which are not in the training data. For a novel image, we obtain its binary code by inputing it into the DDH network and make a forward propagation as below:

$$\mathbf{b}_i=sgn(\mathbf{z}_i)=sgn(\mathbf{W}^T\phi\left(x_i;\theta\right)+\mathbf{c})$$

3 Experiment

Our experiment PC is configured with an Intel(R) Xeon(R) CPU E5–2650 v3 @ 2.30 GHz with 40 cores and the the RAM is 128.0 GB and the GPU is GeForce GTX TITAN X with 12 GB.

3.1 Datasets

We conduct experiments on three challenging datasets, the Oxford 17 Category Flower Dataset, the CIFAR-10 color images, and the NUS-WIDE. We test our binary descriptor on various tasks, including image retrieval and image classification.

1. **CIFAR-10 Dataset** [17] contains 10 object categories and each class consists of 6,000 images, resulting in a total of 60,000 images. The dataset is split into training and test sets, with 50,000 and 10,000 images respectively.
2. **NUS-WIDE dataset** [5] has nearly 270,000 images collected from the web. It is a multi-label dataset in which each image is annotated with one or multiple class labels from 81 classes. Following [19], we only use the images associated with the 21 most frequent classes. For these classes, the number of images of each class is at least 5000. We use 4,000 for training and 1,000 for testing.

3. **The Oxford 17 Category Flower Dataset** [27] contains 17 categories and each class consists of 80 images, resulting in a total of 1,360 images. The dataset is split into the training (40 images per class), validation (20 images per class), and test (20 images per class) sets.

3.2 Results on Image Retrieval

To evaluate the performance of the proposed DDH, we test our method on the task of image retrieval. We compare DDH with other hashing methods, such as LSH [1], ITQ [9], HS [31], Spectral hashing (SpeH) [45], Spherical hashing (SphH) [14], KMH [12], Deep Hashing (DH) [24] and DeepBit [23], Semi-supervised PCAH [44] on the CIFAR-10 dataset and NUS-WIDE. We set the $K_1 = 15$ and $K_2 = 6$ to construct labels, and the learning rate as 0.001, $\lambda_1 = 15$, $\lambda_2 = 0.00001$ and batch-size = 128. Table 2 shows the CIFAR-10 retrieval results based on the mean Average Precision (mAP) of the top 1,000 returned images with respect to different bit lengths, while Table 3 shows the mAP value of NUS-WIDE dataset calculated based on the top 5,000 returned neighbors. The precision/recall in CIFAR-10 dataset is shown in Fig. 3.

Table 2. Performance comparison (mAP) of different unsupervised hashing algorithms on the CIFAR-10 dataset. The mean Average Precision (mAP) are calculated based on the top 1,000 returned images with respect to different number of hash bits.

Method	16 bit	32 bit	64 bit
Method	16 bit	32 bit	64 bit
KMH	0.136	0.139	0.145
SphH	0.145	0.146	0.154
SH	0.130	0.141	0.139
PCAH	0.129	0.126	0.121
LSH	0.126	0.138	0.157
PCA-ITQ	0.157	0.162	0.166
DH	0.162	0.166	0.170
DeepBit	0.194	0.249	0.277
DDH	**0.447**	**0.486**	**0.535**

From these results, we have the following observations:

(1) Our method significantly outperforms the other deep or non-deep hashing methods in all datasets. In CIFAR-10, the improvement of DDH over the other methods is more significant, compared with that in NUS-WIDE dataset. Specifically, it outperforms the best counterpart (DeepBit) by 25.3%, 23.7% and 25.8% for 16, 32 and 64-bit hash codes. One possible reason is that CIFAR-10 contains simple images, and the constructed neighborhood

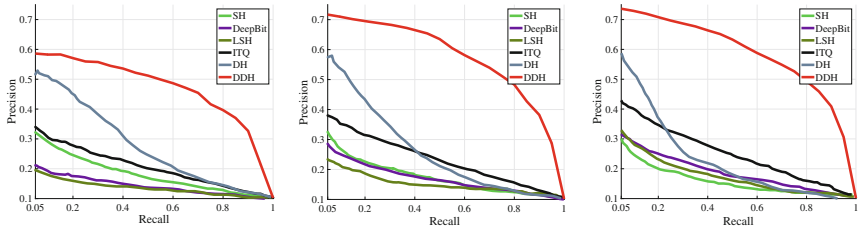
Table 3. Performance comparison (mAP) of different unsupervised hashing algorithms on the NUS-WIDE dataset. The mAP is calculated based on the top 5,000 returned neighbors for NUS-WIDE dataset.

Method	12 bit	24 bit	32 bit	48 bit
CNNH	0.611	0.618	0.625	0.608
FastH	**0.621**	**0.650**	**0.665**	**0.685**
SDH	0.568	0.600	0.608	0.637
KSH	0.556	0.572	0.581	0.588
LFH	0.571	0.568	0.568	0.585
ITQ	0.452	0.468	0.472	0.477
SH	0.454	0.406	0.405	0.400
DDH	**0.675**	**0.680**	**0.701**	**0.712**

structure is more accurate than the other two datasets. DDH improves the state-of-the-arts by 5.4%, 3.0%, 3.6% and 2.7% in NUS-WIDE dataset.

(2) Table 2 shows that DeepBit and FashH are strong competitors in terms of mAP in CIFAR-10 and NUS-WIDE dataset. But the performance gap of DeepBit and our DDH is still very large, which is probably due to that DeepBit uses only 3 fully connected layers to extract the features. Figure 3 shows that most of the hashing methods can achieve a high recall for small number of retrieved samples (or recall). But obviously, our DDH significantly outperforms the others.

(3) With the increase of code length, the performance of most hashing methods is improved accordingly. An exception is PCAH, which has no improvement with the increase of code length.



(a) 16 bits (b) 32 bits (c) 64 bits

Fig. 3. Precision/recall curves of different unsupervised hashing methods on the CIFAR-10 dataset with respect to 16, 32 and 64 bits, respectively

To make fair comparison with the non-deep hashing methods, and validate that our improvement is not only caused by the deep features, we conduct non-deep hashing methods with deep features extracted by the CNN-F pre-trained

on ImageNet. The results are reported in Table 4, where "ITQ+CNN" denotes the ITQ method with deep features and other methods have similar notations. When we run the non-deep hashing methods on deep features, the performance is usually improved compared with the hand-crafted features.

Table 4. Performance comparison (mAP) of different hashing algorithms with deep features on the CIFAR-10 dataset.

Method	12 bit	24 bit	32 bit	48 bit
ITQ + CNN	0.237	0.246	0.255	0.261
SH + CNN	0.183	0.164	0.161	0.161
SPLH + CNN	**0.299**	**0.330**	**0.335**	**0.330**
LFH + CNN	0.208	0.242	0.266	0.339
DDH	**0.414**	**0.467**	**0.486**	**0.512**

By constructing the neighborhood structure using the labels, our method can be easily modified as a supervised hashing method. Therefore, we also compared with supervised hashing methods, and show the mAP results on NUS-WIDE dataset in Table 5. It is obvious that our DDH outperforms the state-of-the-art deep and non-deep supervised hashing algorithms by a large margin, which are 5.7%, 5.8%, 7.8% and 8.1% for 12, 24, 32, and 48-bits hash codes. This indicates that the performance improvement of DDH is not only due to the constructed neighborhood structure, but also the other components.

Table 5. Results on NUS-WIDE. The table shows other deep network with supervised pair-wise labels. The mAP value is calculated based on the top 5000 returned neighbors for NUS-WIDE dataset.

Method	16 bit	24 bit	32 bit	48 bit
DRSCH	**0.618**	**0.622**	**0.623**	0.628
DSCH	0.592	0.597	0.611	0.609
DSRH	0.609	0.618	0.621	**0.631**
DDH	**0.675**	**0.680**	**0.701**	**0.712**

3.3 Results on Object Recognition

In the task of object recognition, the algorithm needs to recognize very similar object (daisy, iris and pansy etc.). So it requires more discriminative binary codes to represent images that look very similar. In this paper, we use the Oxford 17 Category Flower Dataset to evaluate our method on object recognition and we compared with several real-valued descriptors such as HOG [6] and SIFT.

Due to the variation of color distributions, pose deformations and shapes, "Flower" recognition becomes more challenging. Besides, we need to consider the computation cost while one wants to recognize the flowers in the wild using mobile devices, which makes us generate very short and efficient binary codes to discriminate flowers. Following the setting in [27], we train a multi-class SVM classifier with our proposed binary descriptor. Table 6 compares the classification accuracy of the 17 categories flowers using different descriptors proposed in [27], [28], including low-level (Color, Shape, Texture), and high-level (SIFT and HOG) features. Our proposed binary descriptor with 256 dimensionality improves previous best recognition accuracy by around 5.01% (80.11% vs. 75.1%). We also test our proposed method with 64 bits, which still outperforms the state-of-art result (76.35% vs. 75.1%). We also test the computational complexity during SVM classifier training with only costing 0.3 s training on 256 bits and 0.17s on 64 bits. Compared with other descriptors, such as Color, Shape, Texture, HOG, HSV and SIFT, DDH demonstrates its efficiency and effectiveness.

Table 6. The categorization accuracy (mean%) and training time for different features on the Oxford 17 Category Flower Dataset

Descriptors	Accuracy	Time
Colour	60.9	3
Shape	70.2	4
Texture	63.7	3
HOG	58.5	4
HSV	61.3	3
SIFT-boundary	59.4	5
SIFT-internal	70.6	4
DeepBit (256bits)	**75.1**	**0.07**
DDH (64bits)	76.4	0.12
DDH(256bits)	**80.5**	**0.30**

4 Conclusion and Future Work

In this work, we address two central problems remaining largely unsolved for image hashing: (1) how to directly generate binary codes without relaxation? (2) how to equip the binary representation with the ability of accurate image retrieval? We resolve these problems by introducing an intermediate variable and a loss function steering the learning process, which is based on the neighborhood structure in the original space. Experiments on real datasets show that our method can outperform other unsupervised and supervised methods to achieve the state-of-the-art performance in image retrieval and object recognition. In the

future, it is necessary to improve the classification accuracy by incorporating a classification layer at the end of this architecture.

Acknowledgment. This work was supported in part by the National Natural Science Foundation of China under Project 61502080, Project 61632007, and the Fundamental Research Funds for the Central Universities under Project ZYGX2016J085, Project ZYGX2014Z007.

References

1. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM **51**(1), 117–122 (2008)
2. Bay, H., Ess, A., Tuytelaars, T., Gool, L.V.: Speeded-up robust features (SURF). Comput. Vis. Image Underst. **110**(3), 346–359 (2008). Similarity Matching in Computer Vision and Multimedia
3. Cao, Z., Long, M., Wang, J., Yu, P.S.: HashNet: deep learning to hash by continuation. arXiv preprint arXiv:1702.00758 (2017)
4. Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Chua, T.S.: SCA-CNN: spatial and channel-wise attention in convolutional networks for image captioning. arXiv preprint arXiv:1611.05594 (2016)
5. Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: NUS-WIDE: a real-world web image database from national university of Singapore. In: ACM International Conference on Image and Video Retrieval, p. 48 (2009)
6. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR, vol. 1, pp. 886–893. IEEE (2005)
7. Do, T.-T., Doan, A.-D., Cheung, N.-M.: Learning to hash with binary deep neural network. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9909, pp. 219–234. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46454-1_14
8. Fan, B., Wu, F., Hu, Z.: Rotationally invariant descriptors using intensity order pooling. IEEE Trans. Pattern Anal. Mach. Intell. **34**(10), 2031–2045 (2012)
9. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **35**(12), 2916–2929 (2013)
10. Gu, Y., Ma, C., Yang, J.: Supervised recurrent hashing for large scale video retrieval. In: ACM Multimedia, pp. 272–276. ACM (2016)
11. Guo, J., Zhang, S., Li, J.: Hash learning with convolutional neural networks for semantic based image retrieval. In: Bailey, J., Khan, L., Washio, T., Dobbie, G., Huang, J.Z., Wang, R. (eds.) PAKDD 2016. LNCS (LNAI), vol. 9651, pp. 227–238. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-31753-3_19
12. He, K., Wen, F., Sun, J.: K-means hashing: an affinity-preserving quantization method for learning binary compact codes. In: NIPS, pp. 2938–2945 (2013)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
14. Heo, J.P., Lee, Y., He, J., Chang, S.F., Yoon, S.E.: Spherical hashing. In: CVPR, pp. 2957–2964. IEEE (2012)
15. Kang, W.C., Li, W.J., Zhou, Z.H.: Column sampling based discrete supervised hashing. In: AAAI (2016)

16. Kong, W., Li, W.J.: Isotropic hashing. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, vol. 25, pp. 1646–1654. Curran Associates, Inc. (2012)
17. Krizhevsky, A.: Learning multiple layers of features from tiny images (2012)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: NIPS, pp. 1097–1105 (2012)
19. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: CVPR, pp. 3270–3278 (2015)
20. Li, W., Wang, S., Kang, W.: Feature learning based deep supervised hashing with pairwise labels. In: IJCAI, pp. 1711–1717 (2016)
21. Li, X., Shen, C., Dick, A., van den Hengel, A.: Learning compact binary codes for visual tracking. In: CVPR, pp. 2419–2426 (2013)
22. Li, Y., Liu, J., Wang, Y., Lu, H., Ma, S.: Weakly supervised RBM for semantic segmentation. In: IJCAI, pp. 1888–1894 (2015)
23. Lin, K., Lu, J., Chen, C.S., Zhou, J.: Learning compact binary descriptors with unsupervised deep neural networks. In: CVPR, June 2016
24. Liong, V.E., Lu, J., Wang, G., Moulin, P., Zhou, J.: Deep hashing for compact binary codes learning. In: CVPR, pp. 2475–2483, June 2015
25. Brown, M., Hua, G., Winder, S.: Discriminative learning of local image descriptors. IEEE Trans. Pattern Anal. Mach. Intell. (2010)
26. Nguyen, T.V., Sepulveda, J.: Salient object detection via augmented hypotheses. In: IJCAI, pp. 2176–2182 (2015)
27. Nilsback, M.E., Zisserman, A.: A visual vocabulary for flower classification. In: CVPR, vol. 2, pp. 1447–1454. IEEE (2006)
28. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: Sixth Indian Conference on Computer Vision, Graphics & Image Processing. ICVGIP 2008, pp. 722–729. IEEE (2008)
29. Pan, Z., Jin, P., Lei, J., Zhang, Y., Sun, X., Kwong, S.: Fast reference frame selection based on content similarity for low complexity HEVC encoder. J. Vis. Commun. Image Represent. **40**, 516–524 (2016)
30. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
31. Salakhutdinov, R., Hinton, G.: Semantic hashing. Int. J. Approximate Reasoning **50**(7), 969–978 (2009)
32. Song, J., Gao, L., Liu, L., Zhu, X., Sebe, N.: Quantization-based hashing: a general framework for scalable image and video retrieval. Pattern Recognition (2017)
33. Song, J., Gao, L., Nie, F., Shen, H.T., Yan, Y., Sebe, N.: Optimized graph learning using partial tags and multiple features for image and video annotation. IEEE Trans. Image Process. **25**(11), 4999–5011 (2016)
34. Song, J., Gao, L., Puscas, M.M., Nie, F., Shen, F., Sebe, N.: Joint graph learning and video segmentation via multiple cues and topology calibration. In: ACM Multimedia, pp. 831–840 (2016)
35. Song, J., Yang, Y., Yang, Y., Huang, Z., Shen, H.T.: Inter-media hashing for large-scale retrieval from heterogeneous data sources. In: SIGMOD, pp. 785–796 (2013)
36. Song, J., Yang, Y., Huang, Z., Shen, H.T., Luo, J.: Effective multiple feature hashing for large-scale near-duplicate video retrieval. IEEE Trans. Multimedia **15**(8), 1997–2008 (2013)
37. Szegedy, C., Ioffe, S., Vanhoucke, V.: Inception-v4, inception-ResNet and the impact of residual connections on learning. CoRR abs/1602.07261 (2016)

38. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR, pp. 1–9 (2015)
39. Targ, S., Almeida, D., Lyman, K.: Resnet in Resnet: generalizing residual architectures. CoRR abs/1603.08029 (2016)
40. Tian, Q., Chen, S.: Cross-heterogeneous-database age estimation through correlation representation learning. Neurocomputing **238**, 286–295 (2017)
41. Tola, E., Lepetit, V., Fua, P.: Daisy: an efficient dense descriptor applied to wide-baseline stereo. IEEE Trans. Pattern Anal. Mach. Intell. **32**(5), 815–830 (2010)
42. Wan, J., Wu, P., Hoi, S.C.H., Zhao, P., Gao, X., Wang, D., Zhang, Y., Li, J.: Online learning to rank for content-based image retrieval. In: IJCAI, pp. 2284–2290 (2015)
43. Wang, J., Zhang, T., Song, J., Sebe, N., Shen, H.T., et al.: A survey on learning to hash. IEEE Trans. Pattern Anal. Mach. Intell. (2017)
44. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for scalable image retrieval. In: CVPR, pp. 3424–3431 (2010)
45. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Koller, D., Schuurmans, D., Bengio, Y., Bottou, L. (eds.) NIPS, pp. 1753–1760 (2009)
46. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: AAAI, pp. 2156–2162 (2014)
47. Yang, H.F., Lin, K., Chen, C.S.: Supervised learning of semantics-preserving hash via deep convolutional neural networks. IEEE Trans. Pattern Anal. Mach. Intell. (2017)
48. Zhang, P., Zhang, W., Li, W.J., Guo, M.: Supervised hashing with latent factor models. In: SIGIR, pp. 173–182 (2014)