International Journal of Computer Vision (2019) 127:1217–1234
https://doi.org/10.1007/s11263-019-01174-4

Deep Supervised Hashing for Fast Image Retrieval

Haomiao Liu[1,2] · Ruiping Wang[1,2] · Shiguang Shan[1,2] · Xilin Chen[1,2]

Received: 10 August 2017 / Accepted: 6 March 2019 / Published online: 16 March 2019

Abstract
In this paper, we present a new hashing method to learn compact binary codes for highly efficient image retrieval on large-scale datasets. While the complex image appearance variations still pose a great challenge to reliable retrieval, in light of the recent progress of Convolutional Neural Networks (CNNs) in learning robust image representation on various vision tasks, this paper proposes a novel Deep Supervised Hashing method to learn compact similarity-preserving binary code for the huge body of image data. Specifically, we devise a CNN architecture that takes pairs/triplets of images as training inputs and encourages the output of each image to approximate discrete values (e.g. $+1/-1$). To this end, the loss functions are elaborately designed to maximize the discriminability of the output space by encoding the supervised information from the input image pairs/triplets, and simultaneously imposing regularization on the real-valued outputs to approximate the desired discrete values. For image retrieval, new-coming query images can be easily encoded by forward propagating through the network and then quantizing the network outputs to binary codes representation. Extensive experiments on three large scale datasets CIFAR-10, NUS-WIDE, and SVHN show the promising performance of our method compared with the state-of-the-arts.

Keywords Image retrieval · Hashing · Convolutional network · Contrastive loss · Triplet ranking loss

1 Introduction

In recent years, hundreds of thousands of images are uploaded to the Internet every day, making it extremely difficult to find relevant images according to different users' request. For example, content-based image retrieval retrieves images that are similar to a given query image, where "similar" may refer to visually similar or semantically similar. Suppose that both the images in the database and the query image are represented by real-valued features, the most straightforward way of looking for relevant images is by ranking the database images according to their distances to the query image in the feature space, and returning the closest ones. However, for a database with millions or even billions of images, which is quite common nowadays, even a linear search through the database would cost a great deal of time and memory.

To address the inefficiency of real-valued features, hashing approaches are proposed to map images to compact binary codes that approximately preserve the data structure in the original space (Jégou et al. 2011; Liu et al. 2012; Wang et al. 2012), for example. Since the images are represented by binary codes instead of real-valued features, the time and memory costs of searching can be greatly reduced. However, the retrieval performance of most existing hashing methods heavily depends on the features they use, which are basically extracted in an unsupervised manner, thus more suitable for dealing with the visual similarity search rather than the semantic similarity search. On the other hand, recent progress in image classification (Zhang et al. 2016; Krizhevsky et al. 2012; Szegedy et al. 2015), object detection (Szegedy et al. 2013), face recognition (Sun et al. 2014), and many other vision tasks (Deng et al. 2014; Long et al. 2015) demonstrate the impressive learning ability of Convolutional Neural

Communicated by A.W.M Smeulders.

✉ Ruiping Wang
wangruiping@ict.ac.cn

Haomiao Liu
haomiao.liu@vipl.ict.ac.cn

Shiguang Shan
sgshan@ict.ac.cn

Xilin Chen
xlchen@ict.ac.cn

[1] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences (CAS), Beijing 100190, China

[2] University of Chinese Academy of Sciences (UCAS), Beijing 100049, China

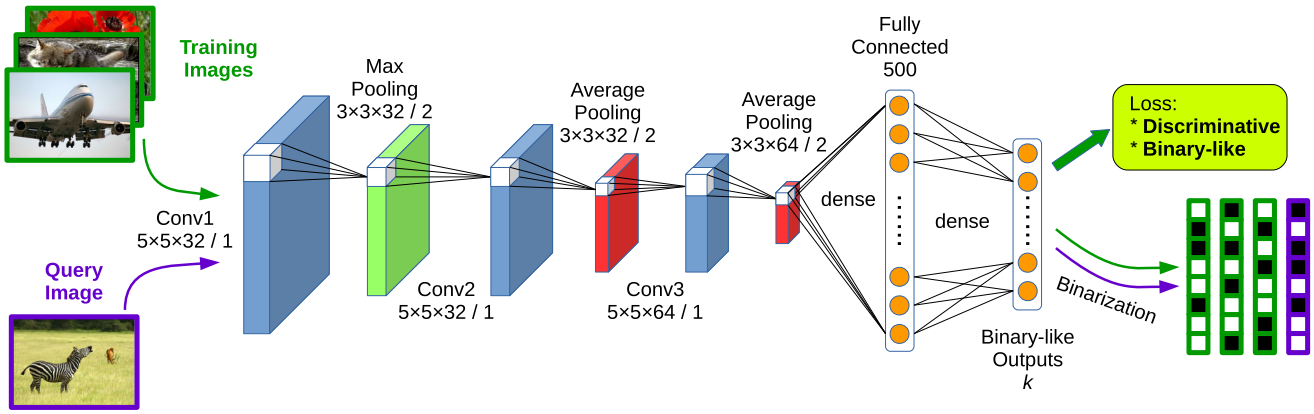

Fig. 1 An example network structure of our method. The network consists of 3 convolution-pooling layers and 2 fully connected layers. The filters in convolution layers are of size 5×5 with stride 1 (32, 32, and 64 filters in the three convolution layers respectively), and pooling over 3×3 patches with stride 2. The first fully connected layer contains 500 nodes, and the second (output layer) has k (the code length) nodes. The loss functions are designed to learn similarity-preserving binary-like codes by exploiting discriminability terms and a regularizer. Binary codes are obtained by quantizing the network outputs of images

Networks (CNNs). In these different tasks, the CNNs can be viewed as a feature extractor guided by the objective functions specifically designed for the individual tasks. The successful applications of CNNs in various tasks imply that the features learned by CNNs can well capture the underlying semantic structure of images in spite of significant appearance variations.

Inspired by the robustness of CNN features, we propose a binary code learning framework by exploiting the CNN structure, named Deep Supervised Hashing (DSH). In our method, first we devise a CNN model which takes image pairs or triplets along with labels indicating the degree of similarity as training inputs, and produces binary codes as outputs, as shown in Fig. 1. In practice, we generate image pairs/triplets online so that many more image pairs/triplets can be utilized in the training stage. The loss functions are designed to pull the network outputs of similar images together and push the outputs of dissimilar ones far away, so that the learned Hamming space can well approximate the semantic structure of images. Ideally, we prefer to directly optimize the binary codes in the Hamming space instead of relaxing them to real values in order to avoid the possible performance degradation caused by quantization loss. However, optimizing the non-differentiable loss function in Hamming space would inevitably incur gradient issues, thus making the optimization of the CNN model difficult. As a trade-off for the aforementioned problem, the network outputs are relaxed to real values, while simultaneously a regularizer is imposed to encourage the real-valued outputs to approach the desired discrete values, as illustrated in Fig. 2. Under this framework, images can be easily encoded by first forward propagating through the network and then quantizing the network outputs to binary codes representation.

Preliminary results of the method have been published in Liu et al. (2016). Compared with the conference version, this

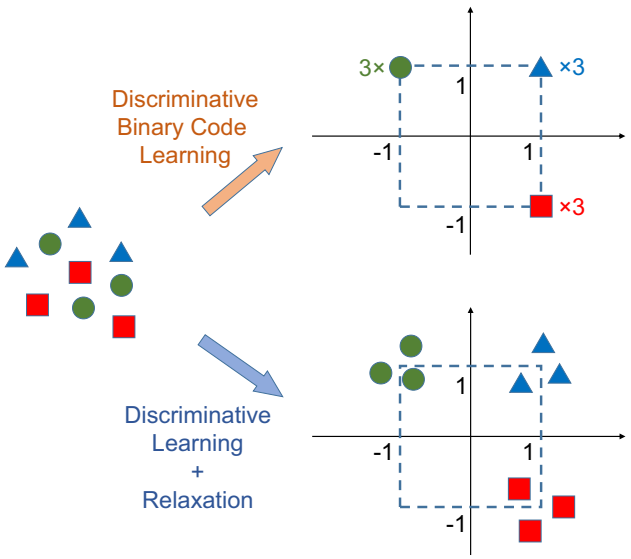

Fig. 2 Illustration of the regularizer proposed in our Deep Supervised Hashing (DSH) method. Ideally, we would like to directly learn discriminative binary codes in the Hamming space, as illustrated in the top right of this figure. However, since directly learning binary codes would inevitably cause gradient issues, optimization of the CNN model would be difficult. To deal with this problem, the binary Hamming space is relaxed to a binary-like Euclidean space by imposing a regularizer on the CNN outputs, as shown in the bottom right of the figure

paper has made three major extensions. First, we generalize the framework to be compatible with not only the previous pair-wise contrastive loss but also the triplet ranking loss, which is also widely adopted in many hash learning methods, e.g. Lai et al. (2015) and Zhao et al. (2015). Second, we provide more detailed comparisons and discussions regarding different strategies for obtaining long binary codes, including training a new model from scratch, fine-tuning a pre-trained model, and concatenating outputs of multiple models that produce short codes. Third, more extensive experiments are

conducted to evaluate each component of the method and compare with more state-of-the-art algorithms on one more challenging dataset SVHN.

The rest of the paper is organised as follows: Sect. 2 discusses the related works to our method. Sections 3 and 4 describe DSH in detail. Section 5 extensively evaluates the proposed method on three large scale datasets. Section 6 gives concluding remarks.

2 Related Work

The problem of nearest neighbor search aims at finding an item from the database, which is the nearest to a query item in terms of a certain metric. In the case that the database is large or that the computation of the distance metric is costly, the computational overhead of exact nearest neighbor search is prohibitively high. As a more practical alternative, approximate nearest neighbor search methods have received more and more attention thanks to their high efficiency (Wang et al. 2017). Specifically, as a representative family of such methods, many hashing algorithms (Gionis et al. 1999; Gong and Lazebnik 2011; Jiang and Li 2017; Kang et al. 2016; Kulis and Darrell 2009; Lai et al. 2015; Li et al. 2016; Lin et al. 2015a; Liu et al. 2017, 2014, 2012; Norouzi and Fleet 2011; Rastegari et al. 2012; Shen et al. 2015; Wang et al. 2012; Weiss et al. 2008; Xia et al. 2014; Zhang et al. 2016; Zhao et al. 2015) have been proposed to boost the performance of approximate nearest neighbor search due to their low time and space complexity. In the early years, researchers mainly focused on data-independent hashing methods, such as a family of methods known as Locality Sensitive Hashing (LSH) (Gionis et al. 1999). LSH methods use random projections to produce hashing bits. It has been proven theoretically that as the code length grows, the Hamming distance between two binary codes asymptotically approaches their corresponding distance in the feature space. However, LSH methods usually require long codes to achieve satisfactory performance, which demands for large amount of memory.

To produce more compact binary codes, data-dependent hashing methods are proposed, which attempt to learn similarity-preserving hashing functions from a training set. These methods can be further divided into unsupervised methods and supervised (semi-supervised) methods. Unsupervised methods only make use of unlabelled training data to learn hash functions. For example, Spectral Hashing (SH) (Weiss et al. 2008) minimizes the weighted Hamming distance of image pairs, where the weights are defined to be the similarity metrics of image pairs in the original feature space; Iterative Quantization (ITQ) (Gong and Lazebnik 2011) tries to minimize the quantization error on projected image descriptors so as to alleviate the information loss caused by the discrepancy between the real-valued feature space and the binary Hamming space.

To better deal with more complicated semantic similarity, supervised methods are proposed to take advantage of label information, such as category labels. CCA-ITQ (Gong and Lazebnik 2011), which is an extension of ITQ, uses label information to find better projections for the image descriptors; Predictable Discriminative Binary Code (DBC) (Rastegari et al. 2012) looks for hyperplanes that separate samples of different categories with large margin as hash functions; Minimal Loss Hashing (MLH) (Norouzi and Fleet 2011) optimizes the upper bound of a hinge-like loss to learn the hash functions. On the other hand, Semi-Supervised Hashing (SSH) (Wang et al. 2012) makes use of the abundant unlabelled data to regularize the hashing functions. While the above methods use linear projections as hashing functions, they can hardly deal with linearly inseparable data. To overcome this limitation, methods such as Supervised Hashing with Kernels (KSH) (Liu et al. 2012) and Binary Reconstructive Embedding (BRE) (Kulis and Darrell 2009) are proposed to learn similarity-preserving hashing functions in kernel space; Deep Hashing (DH) (Erin Liong et al. 2015) exploits a highly non-linear deep network to produce binary codes. While most hashing methods relax the binary codes to real values in optimization and quantize the model outputs to produce binary codes, there is no guarantee that the optimal real-valued codes are still optimal after quantization. To cope with this problem, methods such as Discrete Graph Hashing (DGH) (Liu et al. 2014) and Supervised Discrete Hashing (SDH) (Shen et al. 2015) are proposed to directly optimize the discrete binary codes to overcome the shortcomings of relaxation, and achieve improved retrieval performance.

While the aforementioned hashing methods have certainly achieved success to some extent, they all use hand-crafted features, which cannot well capture the complex semantic information beneath the drastic appearance variations in real-world data and thus limit the retrieval accuracy of the learned binary codes. To tackle this issue, recently several CNN-based hashing methods (Lai et al. 2015; Li et al. 2016; Lin et al. 2015a; Wang et al. 2016; Xia et al. 2014; Zhang et al. 2015; Zhao et al. 2015) are proposed to learn image representations together with binary codes using the promising CNN models. Among them, Lai et al. (2015), Zhang et al. (2015) and Zhao et al. (2015) enforce the network to learn binary-like outputs that preserve the semantic relations of image triplets; Xia et al. (2014) trains a CNN to fit the binary codes computed from the pairwise similarity matrix; Lin et al. (2015a) trains the model with a binary-like hidden layer as features for image classification tasks; most recently, Li et al. (2016) and Wang et al. (2016) propose to impose a regularization term to reduce the discrepancy between the learned real-valued feature space and the desired Hamming space. Cao et al. (2017) proposes an effective scheme by using

an evolving smooth activation function to approximate the non-differentiable sign function, and it shows appealing performance in decreasing the discrepancy between the learned real-valued features and binary codes. While it can generate exactly binary hash codes, the preceding layers may not be sufficiently trained when the non-smoothness of the activation function increases too fast, for the back-propagated gradients decrease as the non-smoothness increases. By coupling image feature extraction and binary code learning, these methods have greatly improved the retrieval accuracies. Nevertheless, there are still some shortcomings with the training objectives of these methods that limit their practical retrieval performance, e.g. the separated hash coding and feature learning steps in Xia et al. (2014) and the incompatibility of the classification loss and the retrieval task in Lin et al. (2015a), which will be detailed in our experiments. In addition, the saturated non-linear activations that most of them (Lai et al. 2015; Lin et al. 2015a; Xia et al. 2014; Zhang et al. 2015; Zhao et al. 2015) employ to approximate the quantization step operate at the risk of possibly slowing down the network training, as indicated by Krizhevsky et al. (2012). To deal with the above problems, we propose to learn the hashing functions in an end-to-end manner by simultaneously exploiting discriminability terms and a quantization regularizer as the loss function of the CNN model.

3 Approach

Our goal is to learn compact binary codes for images such that: (a) similar images should be encoded to similar binary codes in Hamming space, and vice versa; (b) the binary codes of new-coming images could be computed efficiently; (c) an end-to-end learning framework that can simultaneously learn intermediate image representations and binary codes.

Although many hashing methods have been proposed to learn similarity-preserving binary codes, they suffer from the limitations of either hand-crafted features or linear projections. On the other hand, the powerful non-linear models known as CNNs have facilitated the recent successes in computer vision community in various tasks. Based on these observations, we propose to use the CNN models to learn discriminative image representations and compact binary codes simultaneously, which can break out the limitations of both hand-crafted features and linear models. To better illustrate our framework, a simple example network of our method is demonstrated in Fig. 1. Our method first trains the CNN using image pairs or triplets and the corresponding similarity labels. Here the loss functions are elaborately designed to learn similarity-preserving binary-like image representations. Then the CNN outputs are quantized to generate binary codes for new-coming images.

3.1 Loss Function

Let Ω be the RGB image space, our approach aims at learning a mapping from Ω to k-bit binary code: $\mathscr{F} : \Omega \rightarrow \{+1, -1\}^k$, such that similar (either in terms of visual similarity or semantic similarity) images are encoded to similar binary codes and vice versa. For this purpose, the loss function is naturally designed to pull the codes of similar images together, and push the codes of dissimilar images away from each other. In this paper, we consider two typical kinds of loss functions using either pair-wise or triplet supervision, i.e. the contrastive loss that was studied in our preliminary conference version (Liu et al. 2016) and the triplet ranking loss, which is also widely adopted in previous hash learning methods.

3.1.1 Contrastive Loss

The contrastive loss is designed to pull the binary codes of similar images as close as possible and simultaneously push the codes of dissimilar images far away until the distances exceed a pre-set margin, as shown in Fig. 3(a). Formally, for a pair of images $I_1, I_2 \in \Omega$ and the corresponding binary network outputs $\mathbf{b}_1, \mathbf{b}_2 \in \{+1, -1\}^k$, we define $y = 0$ if they are similar, and $y = 1$ otherwise. The loss with respect to the pair of images is defined as:

$$\begin{aligned} L_{con}(\mathbf{b}_1, \mathbf{b}_2, y) = & \frac{1}{2}(1-y) D_h(\mathbf{b}_1, \mathbf{b}_2) \\ & + \frac{1}{2} y \ \max(m_c - D_h(\mathbf{b}_1, \mathbf{b}_2), \ 0) \\ & s.t. \ \ \mathbf{b}_j \in \{+1, -1\}^k, \ \ j \in \{1, 2\} \end{aligned} \tag{1}$$

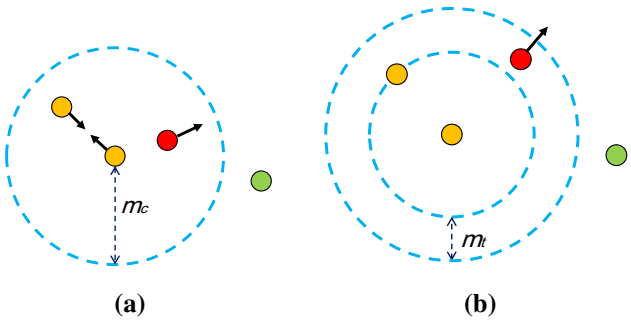

Fig. 3 Illustration of the two kinds of loss functions adopted in our method. **a** The contrastive loss, where similar samples (yellow circles) are encouraged to be as close to each other as possible, while the distances between dissimilar samples (the center yellow circle, the red circle, and the green circle) should be larger than m_c. **b** The triplet ranking loss, where the distance between similar samples (yellow circles) is enforced to be smaller than the distances between dissimilar samples (the center yellow circle, the red circle, and the green circle) by a margin m_t (Color figure online)

where $D_h(\cdot\ ,\ \cdot)$ denotes the Hamming distance between two binary vectors, and $m_c > 0$ is the margin threshold parameter of contrastive loss. The first term in Eq. (1) punishes similar images that are mapped to different binary codes, and the second term punishes dissimilar images mapped to close-by binary codes when their Hamming distance falls below the margin threshold m_c. Here the margin m_c is incorporated to avoid collapsed solution following (Hadsell et al. 2006), that is, only those dissimilar pairs having their distances within a radius are eligible to contribute to the loss function.

Suppose that there are N_c training pairs randomly selected from the training images $\{(I_{i,1}, I_{i,2}, y_i)|i = 1, \ldots, N_c\}$, our goal is to minimize the overall loss function:

$$\begin{aligned}\mathcal{L}_{con} &= \sum_{i=1}^{N_c} L_{con}(\mathbf{b}_{i,1}, \mathbf{b}_{i,2}, y_i) \\ &s.t.\ \ \mathbf{b}_{i,j} \in \{+1, -1\}^k,\ \ j \in \{1, 2\},\ \forall\, i \end{aligned} \tag{2}$$

3.1.2 Triplet Ranking Loss

Similar to the contrastive loss, triplet ranking loss also aims at learning discriminative image representations but in a softer manner. Specifically, unlike the contrastive loss which punishes any non-zero distances between the binary codes of similar images, triplet ranking loss only requires the distances between codes of similar images being smaller than that of dissimilar images, as shown in Fig. 3b. For an image triplet $I_a, I_p, I_n \in \Omega$ and the corresponding binary network outputs $\mathbf{b}_a, \mathbf{b}_p, \mathbf{b}_n \in \{+1, -1\}^k$, where I_a is more similar to I_p than to I_n (here we use the subscripts to denote *anchor*, *positive*, and *negative* respectively). The loss function for this triplet is defined as:

$$\begin{aligned}L_{tri}(\mathbf{b}_a, \mathbf{b}_p, \mathbf{b}_n) = \frac{1}{2} &max(D_h(\mathbf{b}_a, \mathbf{b}_p) \\ &- D_h(\mathbf{b}_a, \mathbf{b}_n) + m_t,\ 0) \\ &s.t.\ \ \mathbf{b}_a, \mathbf{b}_p, \mathbf{b}_n \in \{+1, -1\}^k \end{aligned} \tag{3}$$

where $m_t > 0$ is the margin parameter of the triplet loss. This loss function forces the Hamming distance between $\mathbf{b}_a$ and $\mathbf{b}_n$ to be larger than the Hamming distance between $\mathbf{b}_a$ and $\mathbf{b}_p$ by a margin m_t. Instead of constraining the precise numerical values of the distances, this loss function imposes constraints on the relative distances of all samples to the anchor sample I_a , and is thus considered more suitable for retrieval tasks (Norouzi et al. 2012).

Similarly, suppose that there are N_t training triplets randomly selected from the training images $\{(I_{i,a}, I_{i,p}, I_{i,n}\}|\ i = 1, \ldots, N_t\}$, the overall loss function to minimize is as follows:

$$\begin{aligned}\mathcal{L}_{tri} &= \sum_{i=1}^{N_t} L_{tri}(\mathbf{b}_{i,a}, \mathbf{b}_{i,p}, \mathbf{b}_{i,n}) \\ &s.t.\ \ \mathbf{b}_{i,a}, \mathbf{b}_{i,p}, \mathbf{b}_{i,n} \in \{+1, -1\}^k,\ \forall\, i \end{aligned} \tag{4}$$

3.2 Optimization

It would be preferable if one can directly optimize Eq. (2) or Eq. (4), however it is infeasible because the binary constraints on $\mathbf{b}$ require thresholding the network outputs (e.g. with signum function), and will make it intractable to train the network with standard back propagation algorithm. Although some recent works (Liu et al. 2014; Shen et al. 2015) propose to directly optimize the binary codes, they are not compatible with CNN models. To be specific, due to the memory limitation, CNN models can only be trained with mini-batches, and the optimality of the binary codes produced by such methods can hardly be guaranteed when the batch size is very small compared to the whole training set.

On the other hand, if one totally ignores the binary constraints, it would result in suboptimal binary codes due to the discrepancy between the Euclidean space and the Hamming space. A commonly used relaxation scheme is to utilize *sigmoid* or *tanh* function to approximate the thresholding procedure. Nevertheless, working with such saturated nonlinear functions would inevitably slow down or even restrain the convergence of the network due to gradient vanishing problem (Krizhevsky et al. 2012). To overcome such limitation, in this work we propose to impose a regularizer on the real-valued network outputs to approach the desired discrete values $(+1/-1)$, as shown in Fig. 2. To be specific, we replace the Hamming distance in Eqs. (1) and (3) by Euclidean distance, and impose an additional regularizer to replace the binary constraints, then Eq. (1) is rewritten as:

$$\begin{aligned}L^r_{con}(\mathbf{b}_1, \mathbf{b}_2, y) = &\frac{1}{2}(1-y)||\mathbf{b}_1 - \mathbf{b}_2||_2^2 \\ &+ \frac{1}{2} y\ \max(m_c - ||\mathbf{b}_1 - \mathbf{b}_2||_2^2,\ 0) \\ &+ \alpha(||\ |\mathbf{b}_1| - \mathbf{1}||_1 + ||\ |\mathbf{b}_2| - \mathbf{1}||_1) \end{aligned} \tag{5}$$

where the superscript r denotes the relaxed loss function, $\mathbf{1}$ is a vector of all ones, $||\cdot||_1$ is the L1-norm of vector, $|\cdot|$ is the element-wise absolute value operation, and α is a weighting parameter that controls the strength of the regularizer. Similarly, Eq. (3) is rewritten as:

$$\begin{aligned}L^r_{tri}(\mathbf{b}_a, \mathbf{b}_p, \mathbf{b}_n) = &L^E_{tri}(\mathbf{b}_a, \mathbf{b}_p, \mathbf{b}_n) \\ &+ \alpha(||\ |\mathbf{b}_a| - \mathbf{1}||_1 + ||\ |\mathbf{b}_p| - \mathbf{1}||_1) \\ &+ ||\ |\mathbf{b}_n| - \mathbf{1}||_1) \end{aligned} \tag{6}$$

where $L^E_{tri}(\mathbf{b}_a, \mathbf{b}_p, \mathbf{b}_n) = \frac{1}{2}max(||\mathbf{b}_a - \mathbf{b}_p||_2^2 - ||\mathbf{b}_a - \mathbf{b}_n||_2^2 + m_t, 0)$, and the superscript E denotes squared Euclidean distance.

Here we use L2-norm to measure the distance between network outputs because the subgradients produced by lower-order norms treat the image pairs with different distances equally and thus cannot make use of the information involved in different distance magnitudes. While higher-order norms are also feasible, more computations will be incurred accordingly at the same time. As for the regularizer, L1-norm is chosen rather than higher-order norms for its much less computational cost, which can favorably accelerate the training process.

By substituting Eqs. (5) and (6) into Eqs. (2) and (4) respectively, we rewrite the relaxed overall loss function as follows:

$$\begin{aligned}\mathcal{L}^r_{con} = \sum_{i=1}^{N_c} \Big\{ & \frac{1}{2}(1-y_i)||\mathbf{b}_{i,1} - \mathbf{b}_{i,2}||_2^2 \\ & + \frac{1}{2} y_i \ \max(m_c - ||\mathbf{b}_{i,1} - \mathbf{b}_{i,2}||_2^2, \ 0) \\ & + \alpha(||\ |\mathbf{b}_{i,1}| - \mathbf{1}||_1 + ||\ |\mathbf{b}_{i,2}| - \mathbf{1}||_1) \Big\} \end{aligned} \tag{7}$$

$$\begin{aligned}\mathcal{L}^r_{tri} = \sum_{i=1}^{N_t} \Big\{ & L^E_{tri}(\mathbf{b}_{i,a}, \mathbf{b}_{i,p}, \mathbf{b}_{i,n}) \\ & + \alpha(||\ |\mathbf{b}_{i,a}| - \mathbf{1}||_1 + ||\ |\mathbf{b}_{i,p}| - \mathbf{1}||_1 \\ & + ||\ |\mathbf{b}_{i,n}| - \mathbf{1}||_1) \Big\} \end{aligned} \tag{8}$$

With these objective functions, the network is trained using back-propagation algorithm with mini-batch gradient descent method. To do so, the gradients of Eqs. (7) and (8) w.r.t. $\mathbf{b}$ need to be computed. Since the *max* operations and the absolute value operations in the objective functions are non-differentiable at some certain points, we use subgradients instead, and define the subgradients to be $\mathbf{1}$ at such points. The subgradients of Eqs. (7) and (8) are respectively written as:

$$\begin{aligned}\frac{\partial \mathcal{L}^r_{con}}{\partial \mathbf{b}_{i,j}} = & (-1)^{j+1}(1-y_i)(\mathbf{b}_{i,1} - \mathbf{b}_{i,2}) \\ & + \mathbb{I}\{||\mathbf{b}_{i,1} - \mathbf{b}_{i,2}||_2^2 < m_c\}(-1)^j y_i(\mathbf{b}_{i,1} - \mathbf{b}_{i,2}) \\ & + \alpha\delta(\mathbf{b}_{i,j}) \end{aligned} \tag{9}$$

$$\begin{aligned}\frac{\partial \mathcal{L}^r_{tri}}{\partial \mathbf{b}_{i,a}} &= \mathbb{I}\{L^E_{tri}(\mathbf{b}_{i,a}, \mathbf{b}_{i,p}, \mathbf{b}_{i,n}) > 0\}(\mathbf{b}_{i,n} - \mathbf{b}_{i,p}) + \alpha\delta(\mathbf{b}_{i,a}) \\ \frac{\partial \mathcal{L}^r_{tri}}{\partial \mathbf{b}_{i,p}} &= \mathbb{I}\{L^E_{tri}(\mathbf{b}_{i,a}, \mathbf{b}_{i,p}, \mathbf{b}_{i,n}) > 0\}(\mathbf{b}_{i,p} - \mathbf{b}_{i,a}) + \alpha\delta(\mathbf{b}_{i,p}) \\ \frac{\partial \mathcal{L}^r_{tri}}{\partial \mathbf{b}_{i,n}} &= \mathbb{I}\{L^E_{tri}(\mathbf{b}_{i,a}, \mathbf{b}_{i,p}, \mathbf{b}_{i,n}) > 0\}(\mathbf{b}_{i,a} - \mathbf{b}_{i,n}) + \alpha\delta(\mathbf{b}_{i,n}) \end{aligned} \tag{10}$$

where

$$\delta(x) = \begin{cases} 1, & -1 \le x \le 0 \ or \ x \ge 1 \\ -1, & otherwise \end{cases} \tag{11}$$

is applied element-wisely, and $\mathbb{I}\{condition\} = 1$ when $condition$ is true, and 0 otherwise. With the computed subgradients over mini-batches, the rest of the back-propagation can be done in standard manner.

With such a framework, the binary codes of images can be easily obtained with $sign(\mathbf{b})$. Experiments in Sect. 5.2.2 will validate the advantage of the regularizer over saturated nonlinearities, which are adopted by most existing CNN-based hashing methods (Lai et al. 2015; Lin et al. 2015a; Xia et al. 2014; Zhang et al. 2015; Zhao et al. 2015).

4 Network Structure and Training

4.1 Network Structure

For fast training and thorough evaluation of each component, a relatively shallow CNN structure is designed as illustrated in Fig. 1, which consists of three convolution-pooling layers followed by two fully connected layers. The convolution layers use 32, 32, and 64 5×5 filters with stride 1 respectively, and the pooling is performed over 3×3 windows with stride 2. The first fully connected layer contains 500 nodes, and the second contains k nodes, where k is the length of binary code. All the convolution layers and the first fully connected layer are equipped with the ReLU activation (Nair and Hinton 2010). Note that for real-world applications, any deeper network structures are also compatible with the proposed framework, e.g. AlexNet (Krizhevsky et al. 2012) and ResNet (Zhang et al. 2016). We have evaluations of such different network structure in Sect. 5.2.5, and experiments show that such deeper CNNs can significantly improve the performance of the proposed method.

4.2 Training Methodology

4.2.1 Online Image Pair/Triplet Generation

An intuitive way to train the network is to use the Siamese-like structure as proposed in Hadsell et al. (2006) and feed the network with pre-generated training image pairs/triplets. However, with such a scheme, processing n images could only provide $\frac{n}{2}$ valid image pairs or $\frac{n}{3}$ image triplets, and storing the pre-generated image pairs/triplets could be very space intensive. To make better use of computational resources and storage space, we propose an alternative algorithm that gener-

ates image pairs/triplets online. Specifically, instead of using the Siamese-structure, we adopt the single-trunk CNN structure as illustrated in Fig. 1. With such a structure, we could exploit all the samples in each mini-batch to construct C_n^2 unique pairs or C_n^3 triplets. To cover those image pairs/triplets across batches, in each iteration the training images are randomly selected from the whole training set. By doing so, our method alleviates the need to store the whole pair-wise similarity matrix (or triplet indices), thus being scalable to large-scale datasets. Specifically, for the triplet version of our method, we follow (Hermans et al. 2017) to use the number of triplets that have non-negative loss values to normalize the gradients, which has shown to be beneficial for training triplet loss. Please refer to Hermans et al. (2017) for more details.

As for the margin m_c in the contrastive loss and m_t in the triplet ranking loss, we propose a heuristic strategy by setting $m_c = 2k$ to encourage the codes of dissimilar images to differ at no less than $\frac{k}{2}$ bits, and $m_t = 8$ to constrain the Hamming distance of dissimilar pairs to be larger than that of the similar pairs by at least 2 bits following (Lai et al. 2015).

4.2.2 Schemes for Obtaining Long Codes

To learn models corresponding to different code lengths, if one chose to train each model from scratch, it would be severely wasteful since the preceding layers should have been shared by these models. Besides, as the code length grows, the model would contain more parameters in the output layer, and thus cause a more serious problem, which is known as overfitting. To overcome such limitations, we propose two alternative schemes to obtain long binary codes: (a) **Finetune**. Finetuning has been widely adopted to avoid overfitting on small datasets and has shown promising results, e.g. Lin et al. (2015a, b). In this scheme, we first train a network with a few (e.g. $k = 12$) nodes in the output layer, and then finetune it to obtain the target model with the desired code length. To be specific, during finetuning, only the network parameters in the last layer are trained with a large learning rate, while the parameters in the other layers are trained with a smaller learning rate and thus only change slightly. Therefore, this scheme could relieve overfitting by reducing the number of parameters that need to be learned from scratch. (b) **Ensemble**. This scheme is based on network ensembles, which are widely used in classification tasks (Zhang et al. 2015; Krizhevsky et al. 2012; Szegedy et al. 2015) but nearly unexplored for retrieval tasks. Specifically, we train several networks, each of which has only a few output nodes, and concatenate the outputs of these models to obtain long codes. Since each model contains a relatively small number of parameters, it is less likely to overfit. Moreover, due to random initializations of the individual models, it is possible that models learned with such a scheme could capture complementary information. We will evaluate the training-from-scratch strategy as well as the two alternatives in Sect. 5.2.4, and discuss the advantages and disadvantages of each strategy based on the experimental results.

5 Experiments

5.1 Experimental Settings

Datasets We verify the effectiveness of our proposed method and compare with other state-of-the-art methods (as will be introduced in Sect. 5.3.1) on three widely used datasets: (1) **CIFAR-10** (Krizhevsky 2009). This dataset consists of 60,000 32×32 color images belonging to 10 mutually exclusive categories (6000 images per category). The images are directly used as input for those competing CNN-based methods as well as our DSH. For conventional hashing methods, the images are represented by 512-D GIST descriptors (Oliva and Torralba 2001) following (Liu et al. 2012; Xia et al. 2014). (2) **NUS-WIDE** (Chua et al. 2009). This dataset contains 269,648 images collected from Flickr. The associations between images and 81 concepts are manually annotated. Following (Liu et al. 2012; Xia et al. 2014), we use the images associated with the 21 most frequent concepts, where each of these concepts associates with at least 5000 images, resulting in a total of 195,834 selected images. The images are warped to 64×64 before inputting to the CNN-based methods. For conventional hashing methods, images are represented by the provided 225-D normalized block-wise color moments features. (3) **SVHN** (Netzer et al. 2011). This dataset contains 630,420 32×32 color images of cropped digits (from 0 to 9). The number of samples from each category is highly imbalanced, e.g. there are 13,861 samples of "1" in the training set and only 4659 samples of "9". To overcome this problem, we adopt class-aware sampling (Shen et al. 2016) in the training stage by randomly selecting 20 images from each category in each iteration. For this dataset, we directly use the raw pixels as input for CNN-based methods, and extract 512-D GIST descriptors (Oliva and Torralba 2001) for conventional hashing methods as inputs.

Definition of Similarity In our experiments, similarity labels are defined by semantic-level labels. For CIFAR-10 and SVHN, images from the same category are considered semantically similar, and vice versa. As for image triplets, images that have the same category label as the anchor image are considered as positive, and negative otherwise. For NUS-WIDE, if two images share at least one positive label, they are considered similar, and dissimilar otherwise. On the other hand, if an image share more positive labels with the anchor

image than another image, these three images could form a valid triplet.

Evaluation Protocol On CIFAR-10, the officially provided train/test split is used for experiments, namely, 50,000 images for training the models and 10,000 images for evaluation. On SVHN, all images in the "train" set (73,257 images) are used for training, and 10,000 images randomly selected from the "test" set are used for evaluation (here the "extra" set is ignored for simplicity, and we do not use the whole test set for memory issues). On NUS-WIDE, we randomly sample 10,000 images to form the test query set, and use the rest as training set. Following previous works (Lai et al. 2015; Liu et al. 2012; Xia et al. 2014), the evaluation metrics used are: the mean Average Precision (mAP) for different code lengths, precision-recall curves of 48-bit codes, and mean precision within Hamming radius 2 for different code lengths.

Implementation Details Our DSH method is implemented with the open source platform Caffe[1] (Jia et al. 2014). The weight layers in our models are initialized with "Xavier" initialization (Glorot and Bengio 2010). For training the example network structure illustrated in Fig. 1, the batch size is set to 200, momentum to 0.9, and weight decay to 0.004. Compared to our preliminary publication (Liu et al. 2016), we have changed the learning rate policy on CIFAR-10 and SVHN such that the model is trained with fewer iterations, namely, the initial learning rate is set to 10^{-3} and decreases to 10^{-4} after 60,000 iterations, and to 10^{-5} after 65,000 iterations (70,000 iterations in total). As for NUS-WIDE, since this dataset is larger and more difficult than the other two datasets, the models for this dataset are trained with more iterations (150,000 iterations in total, and the learning rate decreases by 40% after every 20,000 iterations).

5.2 Ablation Study

5.2.1 Training Time

In this part we compare the training time of our DSH method with contrastive and triplet loss on the three datasets. The results are obtained on a server with Titan X GPU and Intel Xeon E5-2620 v3 CPU.

The results (seconds) are shown in Table 1. On all datasets, the models with triplet loss take more training time, since we use all image pairs/triplets in each mini-batch, and the number of triplets is much larger than the number of image pairs. Compared to CIFAR-10, models on NUS-WIDE and SVHN take more training time. For NUS-WIDE, this can be explained by the larger image resolution (64×64 of NUS-WIDE compared to 32×32 of CIFAR-10) and more training iterations (150,000 compared to 70,000). As for SVHN, the image resolution and the training iteration are the same as CIFAR-10, and the difference can be attributed to the batch generation scheme (i.e. the class-aware sampling). Since SVHN have much more training images than CIFAR-10, the random sample selection in each iteration takes more time, resulting in longer training time. While not investigated in our current study, by optimizing the implementation, the sampling time could be significantly reduced and the training could be accelerated.

Table 1 Training time (seconds) of our method with the two different loss functions

Loss	CIFAR-10	NUS-WIDE	SVHN
Contrastive	1376.51	8381.62	9003.90
Triplet	5667.60	16,965.47	17,580.54

5.2.2 Evaluation of the Regularizer

In this part, we validate the effectiveness of the proposed regularizer, and compare it with the standard relaxation scheme used in existing CNN-based hashing methods (Erin Liong et al. 2015; Lai et al. 2015; Xia et al. 2014). Without loss of generality, we only test the case when $k = 12$, and set $m_c = m_t = 24$ in our DSH method according to Sect. 4. We also compare the sigmoid relaxed models with contrastive loss, which are trained almost the same as ours except for using sigmoid function as the activation of the output layer and setting $\alpha = 0$. We test these models with $m_c = \{1, 2, 3, 6\}$ (note that the maximal distance between network outputs of these models is k).

The retrieval mAP of different models with contrastive loss are listed in Table 2. Figure 4 shows the corresponding distribution of network outputs on the test set of CIFAR-10 under different settings (the distributions on other datasets are similar). We make three observations from the comparison results: **First**, without regularization and sigmoid relaxation ($\alpha = 0$), the network outputs concentrate on the quantization threshold 0 (Fig.4a), thus it is likely that neighboring points in the output space are quantized to very different binary codes; **Second**, imposing the regularizer ($\alpha = \{0.001, 0.01, 0.1\}$, Fig.4b,c,d) can reduce the discrepancy between the real-valued output space and the Hamming space, and the retrieval performances can be improved significantly when setting α under a reasonable range (e.g. [0.001, 0.01]); **Third**, with proper settings of the margin parameter m_c, the sigmoid relaxed model can learn binary-like outputs (Fig.4e,f,g). Nonetheless, the retrieval performances of such codes are much inferior to our best-performing ones and are sensitive to m_c. Increasing the number of training iterations

[1] The source code of our DSH with running samples are available at http://vipl.ict.ac.cn/resources/codes or https://github.com/lhmRyan/deep-supervised-hashing-DSH.

Table 2 Retrieval performance (mAP) of models with different loss functions, α, relaxation, and margin

Models	CIFAR-10	NUS-WIDE	SVHN
Regularizer-α-0	0.5607	0.5076	0.8464
Regularizer-α-0.001	0.6571	0.5341	0.8671
Regularizer-α-0.01	0.6778	0.5604	0.8846
Regularizer-α-0.1	0.3674	0.4493	0.8094
Sigmoid-m-6	0.1706	0.4876	0.1200
Sigmoid-m-3	0.3267	0.5067	0.5750
Sigmoid-m-2	0.3327	0.4838	0.5748
Sigmoid-m-1	0.2395	0.4638	0.4548
Regularizer-tri-α-0	0.5626	0.4976	0.7327
Regularizer-tri-α-0.001	0.6360	0.5629	0.7867
Regularizer-tri-α-0.01	0.6067	0.5395	0.8297

The top panel corresponds to the models with contrastive loss and different α. The middle panel shows the models with sigmoid relaxation and different margin. The bottom panel demonstrates the models with triplet loss and different α. The results are obtained with 12-bit binary codes

and carefully tuning m_c might improve the performance of the sigmoid relaxed models, however, it would take much more time to obtain a satisfactory model. Based on the above observations, we empirically set $\alpha = 0.01$ in the following experiments.

On the other hand, our proposed regularizer can also be combined with triplet ranking loss, and the results are shown in Table 2. We can see that ignoring the binary characteristic of the desired output space in the training stage (setting $\alpha = 0$) severely deteriorates the retrieval performance of the learned models (compared to the models trained with $\alpha > 0$), indicating that the proposed regularizer is compatible with multiple kinds of loss functions (contrastive loss and triplet loss in our method). Carefully tuning the hyperparameters (α and m_t) might further improve the performances, however, that is beyond the scope of this paper.

By comparing models with the two loss functions, we can observe that models with contrastive loss outperform those with triplet loss on the single-label datasets (CIFAR-10 and SVHN), and the two groups of models produce similar results on multi-label dataset (NUS-WIDE). A possible explanation for this phenomenon is that contrastive loss has stricter constraints on similar pairs than triplet loss, i.e. the distances between similar pairs should be as small as possible in contrastive loss, whereas the distances between similar pairs are only required to be smaller than the distances between dissimilar pairs in triplet loss. As a result, samples from the same category are more compact when using contrastive loss, resulting in better performance on CIFAR-10 and SVHN. On the other hand, since NUS-WIDE is a multi-label dataset and the similarity relationships are more complex (note that it is possible in NUS-WIDE that both images A and C are similar to image B, but A is dissimilar to C), the contrastive loss might face with some contradictory constraints in optimization (e.g. $D(A, B) = 0$, $D(B, C) = 0$, and $D(A, C) > m_c$ as in the above example), which could possibly increase the difficulty of optimization and thus degrade the quality of the learned model.

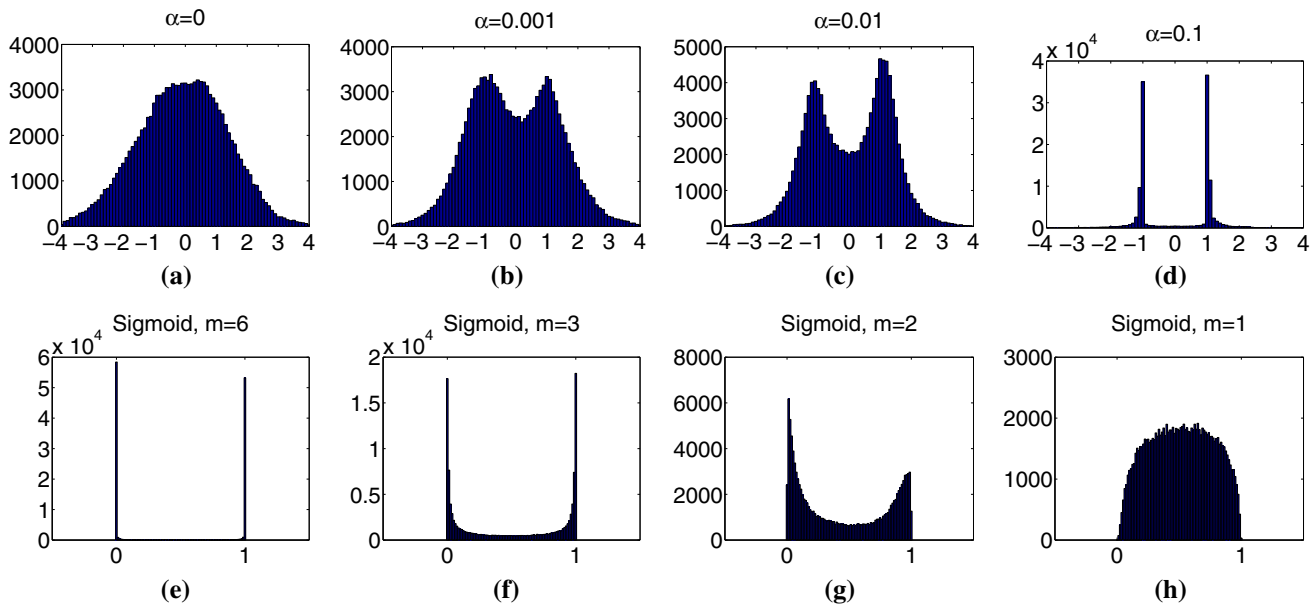

Fig. 4 Distribution of network outputs (with contrastive loss) on the test set of CIFAR-10. **a**-**d** the models using our proposed regularizer under different settings of α, **e**–**h** the sigmoid relaxed models under different settings of m_c

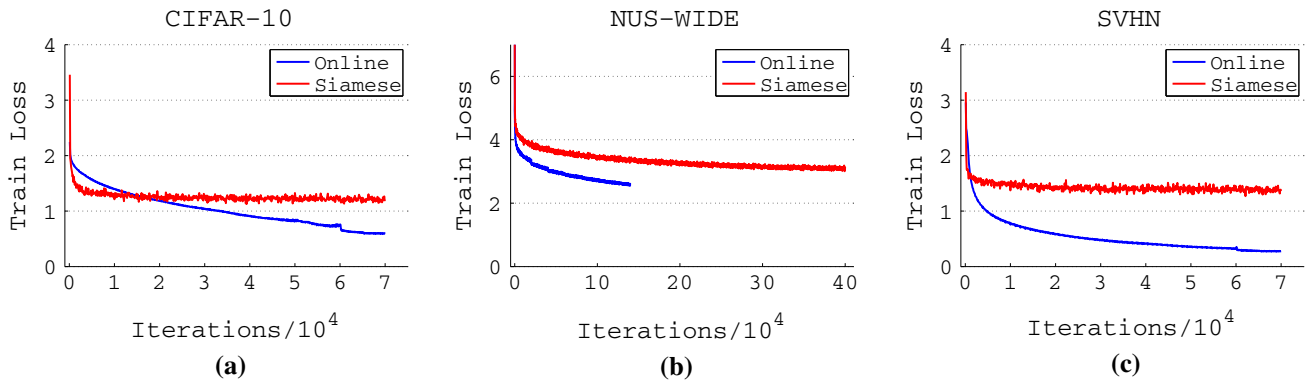

Fig. 5 Comparison of training loss between our online image pair generation scheme and the Siamese alternative. The results on CIFAR-10, NUS-WIDE, and SVHN are shown in (**a**–**c**) respectively

Based on the above results of training time and retrieval performance, in the following experiments, we only give the results of our DSH method with contrastive loss (with $\alpha = 0.01$ and $m_c = 2k$) unless otherwise specified.

5.2.3 Online Versus Offline Image Pair Generation

This part compares the convergence behavior of our online image pair generation scheme against the alternative Siamese scheme, as described in Sect. 4.2.1. Both schemes employ the same network structure and hyperparameters as detailed in previous sections (i.e. $k = 12$ and $m_c = 24$). Due to limited storage space, 10 million image pairs were generated offline for the Siamese scheme, and the learning rate policy is tuned accordingly. For fair comparison, we sample the same number of images for both schemes in each iteration so that the computation costs of the two schemes are approximately the same (since the computations mainly take place in the convolution-pooling layers). To be specific, for contrastive loss with a batch size of 200, about 20,000 image pairs are adopted in each iteration in our online scheme, and the total number of image pairs is about 1.4 billion (with duplicates) for CIFAR-10 and SVHN and 3 billion (with duplicates) for NUS-WIDE. In contrast, 200 images only contribute 100 image pairs for the alternative Siamese scheme in each iteration, and the total number of image pairs is only 10 million in the offline scheme.

Figure 5 shows the training loss against the number of iterations on the three datasets. As can be seen, our online training scheme converges faster than the Siamese alternative, since our online scheme has the capacity to utilize much more image pairs in each iteration, which offers more information about the semantic relations between different images. Besides, by sampling from the whole training set in each iteration, our scheme can make use of more image pairs than the offline generated 10 million pairs for Siamese, and thus satisfactorily converges to a lower loss.

5.2.4 Different Schemes for Generating Long Binary Codes

As mentioned in Sect. 4.2.2, if the last fully connected layer contains a large number of nodes (i.e. more learnable parameters), training the model from scratch may lead to severe overfitting. Based on this assumption, in this part we compare three schemes for obtaining long binary codes, including the training-from-scratch scheme and the two alternatives introduced in Sect. 4.2.2.

Specifically, we first train four models with randomly initialized weights that produce {12, 24, 36, 48}-bit binary codes respectively (the first four rows in Table 3, denoted by "Trained From Scratch"). Then we replace the last fully connected layer of the above 12-bit model with a randomly initialized larger one, and finetune it to get another group of {24, 36, 48}-bit models (the middle three rows in Table 3, denoted by "Finetune"). For finetuning, the learning rate is set to 10^{-3} for the last fully connected layer and 10^{-4} for the preceding layers, and decrease by a factor of 0.6 after every 4,000 iterations. The model is finetuned for 30,000 iterations in total. In addition, three more 12-bit models are trained, resulting in four 12-bit models altogether. Then we concatenate the outputs of these 12-bit models to obtain {24, 36, 48}-bit codes, and report the best performances among their combinations (the last three rows in Table 3, denoted by "Ensemble").

The retrieval mAPs on three datasets are listed in Table 3. It can be found that as the code length grows, the retrieval performances of finetuned models consistently improve, while the performance of models trained from scratch falls, especially on the NUS-WIDE dataset with a large drop. To take a closer look at the situation, we analyze the training/test loss on two sets of models, namely, the 48-bit models trained from scratch, and the finetuned 48-bit models. Figure 6 shows the loss against the number of iterations for the two sets of models. It is clear that the second set of models (finetuned) always converge to lower losses than the first set of models (trained from scratch). Especially on the smallest dataset

Table 3 Comparison of retrieval performance (mAP) of the models trained from scratch, the finetuned models, and the ensemble of multiple models

	Code length	CIFAR-10	NUS-WIDE	SVHN
Trained from scratch	12	0.6778	0.5604	0.8846
	24	0.7046	0.5543	0.8763
	36	0.7024	0.5229	0.8903
	48	0.6906	0.4896	0.8867
Finetune	24	0.7129	0.5780	0.8992
	36	0.7245	0.5812	0.9028
	48	0.7319	0.5875	0.9062
Ensemble	24	0.7282	0.5794	0.9037
	36	0.7458	0.5870	0.9116
	48	0.7536	0.5882	0.9146

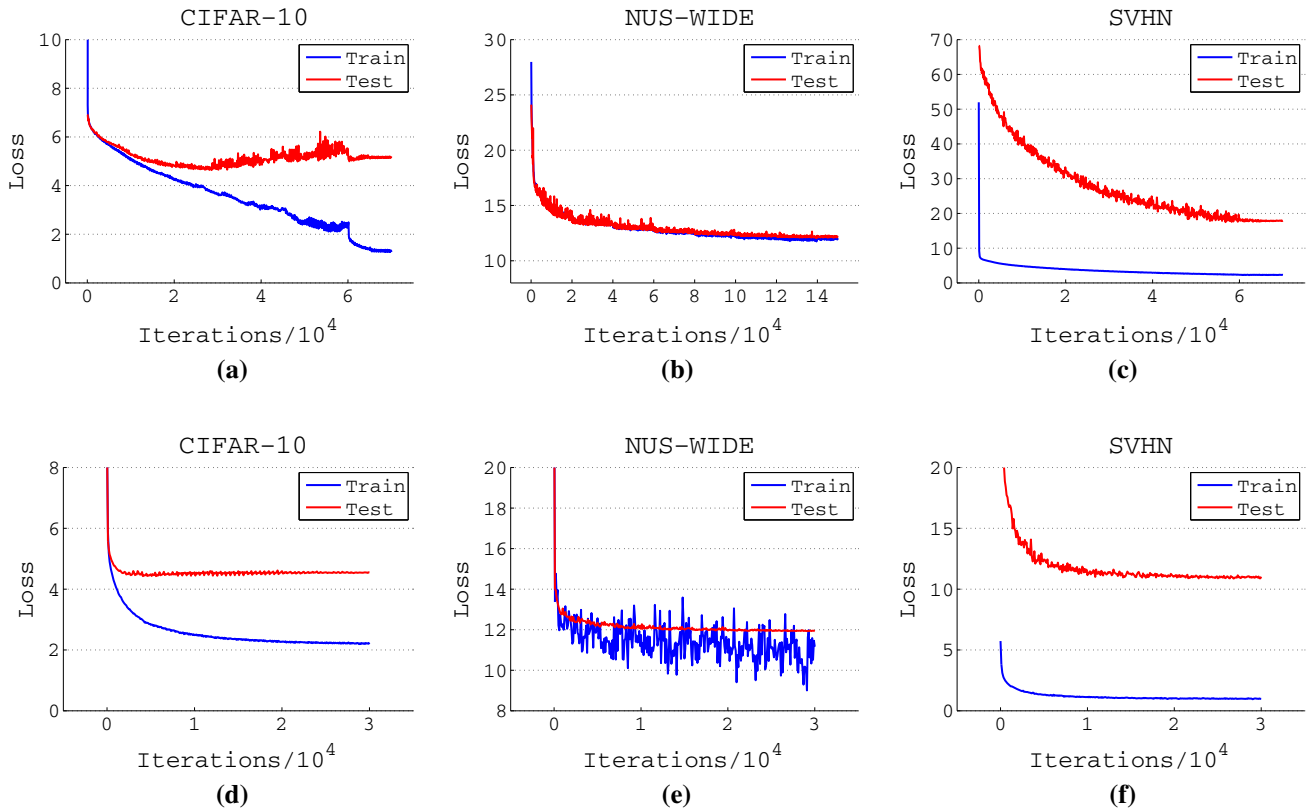

Fig. 6 Comparison of models trained from scratch (**a**–**c**) and finetuned models (**d**–**f**) in terms of training/test loss, on all three datasets

CIFAR-10, on the first model (Fig. 6a) the training loss keeps decreasing, while the test loss decreases as expected at the beginning but starts to increase after about 20,000 iterations, indicating overfitting on the training set. In contrast, on the second model (Fig. 6d), the test loss decreases at first and then favorably stabilizes after only a few thousand iterations. Such observations suggest that the different models with various code lengths can share those preceding layers to reduce training cost as well as to alleviate overfitting.

On the other hand, the model ensemble scheme consistently performs the best for all compared code lengths. Under the same code length, the ensemble codes further improve the retrieval performance of the finetuned codes by about 0.02 in terms of mAP, verifying the effectiveness of network ensembles in retrieval task. In fact in our experiments, not only the best-performing ensemble but also nearly all ensembles we have experimented with can improve the performance. One possible explanation is that the multiple networks can capture complementary image characteristics due to random initialization. Besides, since each individual model in the ensemble only contains a small number of learnable parameters, it is less likely to overfit on the training set. Nevertheless, since

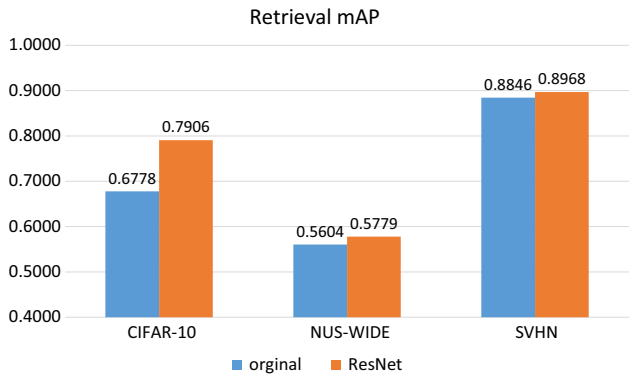

Fig. 7 Retrieval performance (mAP) of our DSH method with the original network structure in Liu et al. (2016) and ResNet (Zhang et al. 2016)

exploiting network ensembles will lead to multiple times of training and encoding cost, we adopt the finetuned models in the following experiments for efficiency consideration.

5.2.5 Deeper Network Structure

As mentioned in Sect. 4.1, our framework is relatively general, and can thus exploit more complex network structures to further boost the performance. In this subsection, we replace the shallow network in the previous sections with the 32-layer ResNet described in Zhang et al. (2016). We adopt the learning rate policy in the original publication, and experiment on the 12-bit binary codes. The results are shown in Fig. 7. It is obvious that incorporating the ResNet structure improves the performance of our DSH method on all datasets, especially on CIFAR-10, confirming that our method can benefit from more advanced model designs. As for the other datasets, the improvements are not as significant. A possible explanation is that the contrastive loss is not well suited for the more complex semantic similarities on NUS-WIDE, and the network structure of the 32-layer ResNet is designed for the 32×32 inputs of CIFAR-10 rather than the 64×64 inputs of NUS-WIDE. On the other hand, the performance improvement of ResNet is only marginal on SVHN, which can be largely attributed to the fact that the performance on this dataset is already nearly saturated with the original shallow network, thus can hardly be further improved.

5.3 Comparison with State of the Arts

5.3.1 Quantitative Analysis

Comparative Methods We compare our method with LSH (Gionis et al. 1999), SH (Weiss et al. 2008), ITQ (Gong and Lazebnik 2011), CCA-ITQ (Gong and Lazebnik 2011), MLH (Norouzi and Fleet 2011), BRE (Kulis and Darrell 2009), KSH (Liu et al. 2012), CNNH (Xia et al. 2014), DLBHC (Lin et al. 2015a), DNNH (Lai et al. 2015), and DPSH (Li et al. 2016). These methods are all implemented using source codes provided by the authors. For fair comparison, all the CNN-based methods, including CNNH, DLBHC, DNNH, DPSH, and DSH, use the same network structure (i.e. the same convolution-pooling layers and the first fully connected layer), as described in Sect. 4.1. Note that while more complicated network structures can be also feasible, we choose to work with a relatively simple one for easier evaluation. For all conventional hash learning methods, they use the hand-crafted features commonly exploited on each dataset, as described in Sect. 5.1.

Training Set We use the whole training set to train models for all methods if possible. However, due to the huge amount of memory demanded by MLH, KSH and CNNH ($O(N^2)$, where N is the number of training images), in our experiments, we randomly select a 20,000 subset from each dataset to train models for these three methods, which already costs more than 10GB of memory.

Parameter Settings The parameters of those comparative methods are set based on the authors' suggestions in the original publications. In particular, we find the divide-and-encode structure devised in DNNH (Lai et al. 2015) largely degrade the retrieval mAP on CIFAR-10 (about 0.07) and bring marginal improvement on NUS-WIDE (0.01–0.03) in our experiments, thus we report the performances of the fully connected version for simplicity.

Results The comparisons of our method against the others are shown in Table 4 and Fig. 8. In general, those CNN-based methods not surprisingly outperform the conventional hash learning methods on all datasets by a large margin, validating the advantage of learning image representations over using hand-crafted features. Among the CNN-based methods, it is observed that our DSH achieves state-of-the-art retrieval accuracy on all datasets. The performance gaps between these methods mainly come from the differences in their training objectives: CNNH trains the model to fit the pre-computed discriminative binary codes. However, as the binary code generation and the network learning are isolated, a mismatch exists between the two stages; DLBHC trains the model with a binary-like hidden layer as features for classification tasks, thus encoding dissimilar images to similar binary codes would not be punished as long as the classification accuracy is unaffected; while DNNH uses triplet-based constraints (rather than the pairwise constraints adopted by our method) to model more complex semantic relations, training its network becomes more difficult, due to the sigmoid nonlinearity and the triplet ranking loss itself. As a result, DNNH performs inferior to our DSH method, especially on CIFAR-10 and SVHN, where the triplet-based constraints cannot provide more information than the pairwise ones since the

Table 4 Comparison of retrieval mAP of our DSH method and the other hashing methods on CIFAR-10, NUS-WIDE, and SVHN

Method	CIFAR-10				NUS-WIDE				SVHN			
	12-bit	24-bit	36-bit	48-bit	12-bit	24-bit	36-bit	48-bit	12-bit	24-bit	36-bit	48-bit
LSH (Gionis et al. 1999)	0.1277	0.1367	0.1407	0.1492	0.3329	0.3392	0.3450	0.3474	0.1363	0.1429	0.1522	0.1537
SH (Weiss et al. 2008)	0.1319	0.1278	0.1364	0.1320	0.3401	0.3374	0.3343	0.3332	0.1677	0.1656	0.1642	0.1615
ITQ (Gong and Lazebnik 2011)	0.1080	0.1088	0.1117	0.1184	0.3425	0.3464	0.3522	0.3576	0.1801	0.1414	0.1383	0.1389
CCA-ITQ (Gong and Lazebnik 2011)	0.1653	0.1960	0.2085	0.2176	0.3874	0.3977	0.4146	0.4188	0.5717	0.6378	0.6470	0.6639
MLH (Norouzi and Fleet 2011)	0.1844	0.1994	0.2053	0.2094	0.3829	0.3930	0.3959	0.3990	0.2951	0.3363	0.3707	0.4140
BRE (Kulis and Darrell 2009)	0.1589	0.1632	0.1697	0.1717	0.3556	0.3581	0.3549	0.3592	0.1408	0.1446	0.1527	0.1517
KSH (Liu et al. 2012)	0.2948	0.3723	0.4019	0.4167	0.4331	0.4592	0.4659	0.4692	0.4940	0.5625	0.5942	0.6212
CNNH (Xia et al. 2014)	0.5425	0.5604	0.5640	0.5574	0.4315	0.4358	0.4451	0.4332	0.7930	0.8004	0.8171	0.8120
DLBHC (Lin et al. 2015a)	0.5503	0.5803	0.5778	0.5885	0.4663	0.4728	0.4921	0.4916	0.8473	0.8706	0.8839	0.8872
DNNH (Lai et al. 2015)	0.6122	0.6270	0.6271	0.6136	0.5617	0.5635	0.5592	0.5630	0.8169	0.8383	0.8371	0.8423
DPSH (Li et al. 2016)	0.6561	0.6549	0.6602	0.6623	0.5647	0.5765	0.5880	0.5886	0.8010	0.8044	0.8247	0.8250
DSH	0.6778	0.7129	0.7245	0.7319	0.5604	0.5780	0.5812	0.5875	0.8846	0.8992	0.9028	0.9062

images only have category labels; As the most similar method to ours, DPSH performs inferior to our method on CIFAR-10 and SVHN, yet marginally better than ours on NUS-WIDE. The differences mainly come from the finetuning scheme (Sect. 5.2.4), the class-aware sampling strategy, and different ratios between the numbers of similar and dissimilar image pairs on the three datasets. Among the above three factors, the first two can explain why DSH performs much better on CIFAR-10 (especially with long codes) and the imbalanced SVHN dataset. On the other hand, since NUS-WIDE dataset has more similar image pairs than the other two datasets (Cao et al. 2017), and DPSH adopts column sampling strategy over the whole training set in the training stage while our method only computes the loss within a single mini-batch, the difference in ratios has larger impact on our method.

To further investigate the performance of conventional hashing methods with CNN features, we test the performance of 48-bit codes generated by LSH, CCA-ITQ, and KSH on CIFAR-10 using two kinds of CNN features, i.e. the L2-normalized 500-D network activations of the first fully-connected layer extracted from (a) **CNN-ours**: our 12-bit model, and (b) **CNN-cls**: a model with the same preceding layers as our model but trained for classification task (obtained by replacing the output layer of our model with a 10-way softmax loss layer), which achieves 80.15% accuracy on the CIFAR-10 test set. Results are shown in Table 5. The performances of conventional methods improve significantly with CNN features (even comparable to our method), and the features from our model are superior to the ones from the classification model, validating again our motivation of learning binary codes in an end-to-end manner. Besides, the non-linear hashing method KSH performs much better than the linear method CCA-ITQ when using GIST features, and as the features become more suitable for retrieval task (from GIST to CNN-cls and to CNN-ours), the performances of both methods are improved and the gap is narrowed down, suggesting that linear hashing methods could be as good as non-linear methods when equipped with suitable image features.

5.3.2 Qualitative Analysis

In this part, we show some failed retrieval cases on CIFAR-10 and NUS-WIDE. We compare the result of the proposed DSH method against LSH (Gionis et al. 1999), ITQ (Gong and Lazebnik 2011), CCA-ITQ (Gong and Lazebnik 2011), MLH (Norouzi and Fleet 2011), KSH (Liu et al. 2012), DLBHC (Lin et al. 2015a), and DNNH (Lai et al. 2015). The results on the two datasets (CIFAR-10 and NUS-WIDE) are shown in Figs. 9 and 10 respectively. For space limitation, only the top-10 feedbacks corresponding to each query image are shown here, with the true matches bounded by red boxes. The number of true matches of each method is

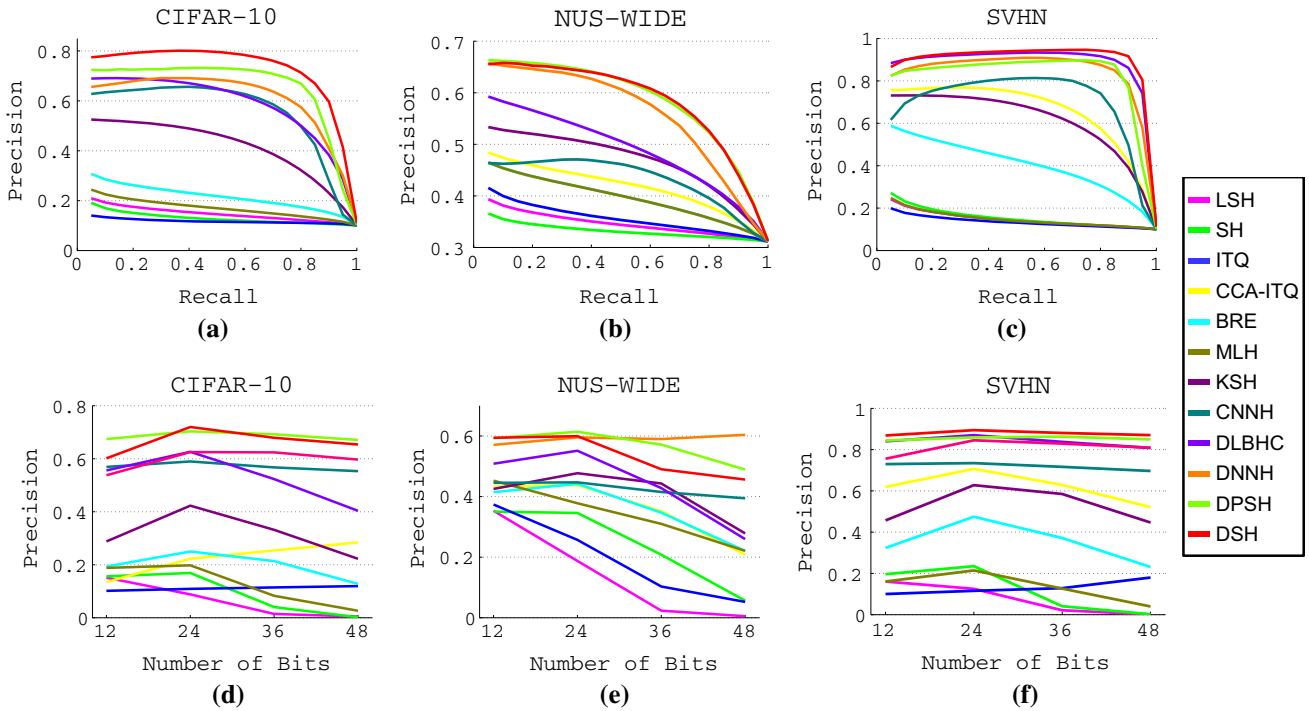

Fig. 8 Comparison of retrieval performance of our DSH method and the other hashing methods on CIFAR-10, NUS-WIDE, and SVHN. **a–c** PR curves (with 48-bit codes). **d–f** Mean precision within Hamming radius 2

Table 5 Retrieval mAP on CIFAR-10 with 48-bit binary codes

	LSH	CCA-ITQ	KSH	DSH
Hand	0.1492	0.2176	0.4167	–
CNN-cls	0.3686	0.6384	0.6978	–
CNN-ours	0.3918	0.7118	0.7014	0.7319

The compared conventional hashing methods are trained with different features, including **Hand**: hand-crafted features (i.e. 512-D GIST features, as reported in Table 4), **CNN-cls**: CNN features from the classification model, and **CNN-ours**: CNN features from our 12-bit model

listed beside the retrieval results, and the label(s) of query images are also provided. We can observe that in the failed cases, although the top feedbacks of our method are dissimilar with the query image, the top feedbacks themselves are similar to each other. Specifically, when compared to the conventional hashing methods, the top feedbacks of CNN-based methods are less diverse, e.g. the top-10 feedbacks of MLH in Fig. 9 contain images of five categories (truck, boat, bird, car, airplane, and horse), while the top-10 feedbacks of our DSH method contains only two categories (airplane and boat). This observation suggests that our method is able to preserve the similarity of images, yet sometimes incorrectly recognizes the semantic meanings of query images, which could be improved by exploiting deeper and more complex network structures.

5.3.3 Comparison of Encoding Time

In real-world applications, generating binary codes for new-coming images should be fast. In this part, we compare the encoding time of our DSH method and 8 other supervised hashing methods: CCA-ITQ (Gong and Lazebnik 2011), MLH (Norouzi and Fleet 2011), BRE (Kulis and Darrell 2009), KSH (Liu et al. 2012), CNNH (Xia et al. 2014), DLBHC (Lin et al. 2015a), DNNH (Lai et al. 2015), and DPSH (Li et al. 2016), including the linear and non-linear conventional hashing methods along with the state-of-the-art CNN-based methods. For thorough comparison, we report the encoding time of CNN-based methods both on CPU and GPU, and the feature extraction time for conventional hashing methods (using the publicly available code of GIST feature extraction, Oliva and Torralba 2001). Since we use the authors' provided features for NUS-WIDE and only extract exactly the same features for CIFAR-10 and SVHN, all comparisons are conducted on CIFAR-10. Without loss of generality, we only report the timings of 24-bit and 48-bit codes. The binary codes of all CNN-based methods are generated with the same version of Caffe. The experiments are carried out on a PC with Intel i7-4770, 32GB RAM, and NVIDIA Titan Black with CUDA-7.5 and cuDnn v4.0.

The logarithmic encoding time (in microseconds, base 10) of such hashing methods is shown in Fig. 11, where results are obtained by averaging over the whole test set. CNN-based

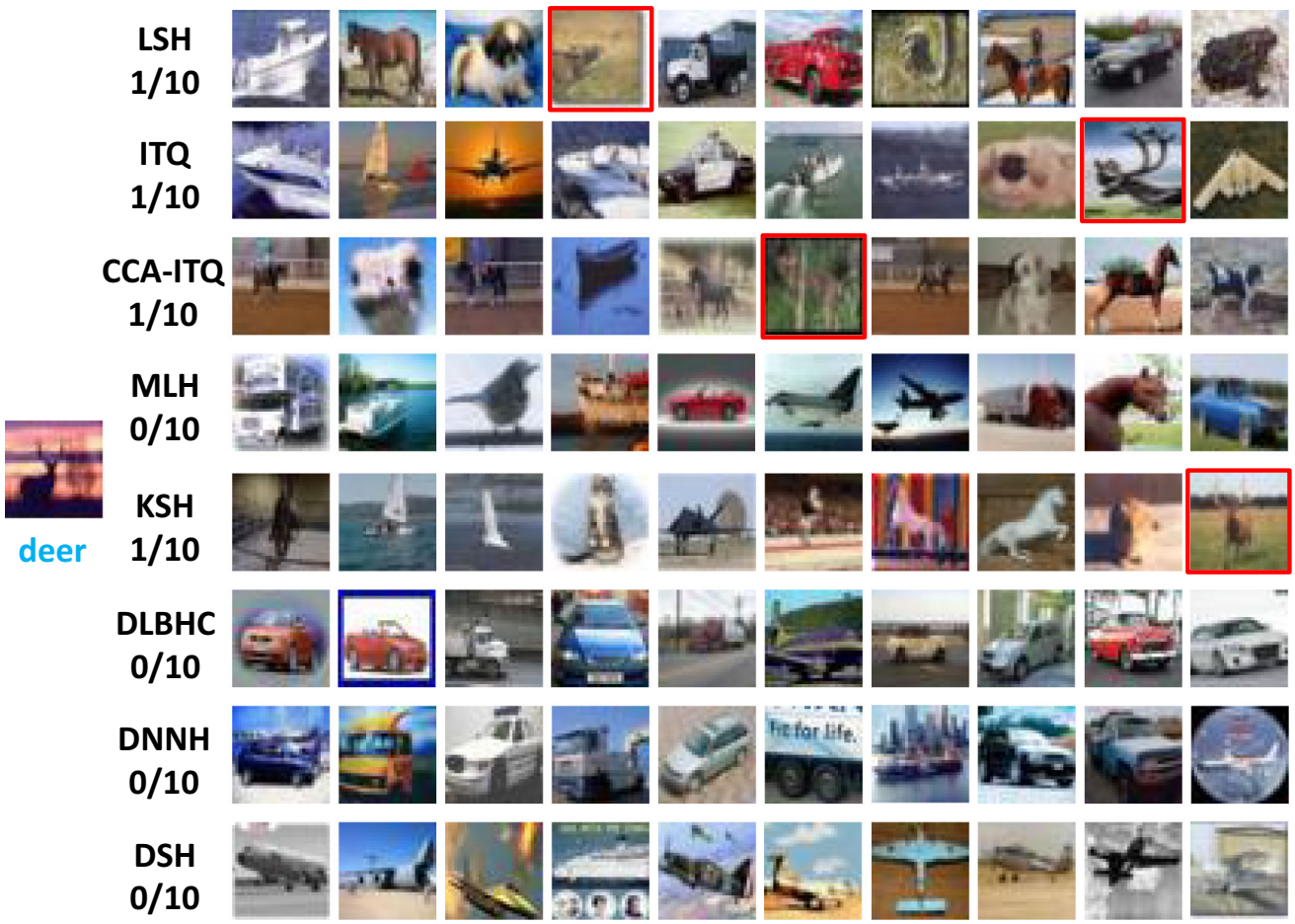

Fig. 9 A failed retrieval case on CIFAR-10, only the top-10 feedbacks are shown due to space limitation. Results are obtained with 48-bit binary codes. Images with red boxes are true matches (Color figure online)

methods take almost the same time to encode a single image with varying code lengths, since the computations mainly take place in the common preceding layers. In general, when only considering generating binary codes from model inputs, even the GPU accelerated version of CNN-based methods are slower than the conventional methods by at least an order of magnitude. However, taking the feature extraction time into consideration, the CNN-based methods are more than 10 times faster than the comparative conventional hashing methods. Moreover, the conventional hashing methods usually require several types of features to achieve comparable retrieval performances to CNN-based methods, which further slows down the whole encoding procedure.

5.3.4 Cross-Dataset Retrieval

It is desirable that models trained on one dataset can be used for accurate image retrieval on another dataset. For this purpose, we conduct experiments in this part to see how the comparative methods and our method perform in such cross-dataset retrieval setting. To this end, we train 12-bit models on the large-scale ImageNet dataset and test the retrieval accuracies on the two object-centric datasets CIFAR-10 and NUS-WIDE. Specifically, we adopt the AlexNet structure, and replace the "fc8" layer with a fully connected layer with 12 nodes. The network is initialized with the model weights pre-trained on ImageNet classification task. To train the model, we set the learning rate to 0.001 for the newly added layer and 0.0001 for the preceding layers. The model is trained using SGD with momentum 0.9 for 60,000 iterations. Triplet loss with $m_t = 8$ and $\alpha = 0.01$ is adopted as supervision signals for a direct comparison with DNNH. After the training finishes, we test the model with exactly the same protocol as the above sections. For comparison, we compare the performance with four representative hashing methods, i.e. LSH, ITQ, CCA-ITQ and DNNH, to compare the relative performance of our method. For non-deep methods (LSH, ITQ, and CCA-ITQ), we extract "fc7" features of the training and test set of the two datasets using the pre-trained AlexNet. For DNNH, we fine-tune the model similarly as ours. All comparative methods are implemented with the source codes released by the original authors.

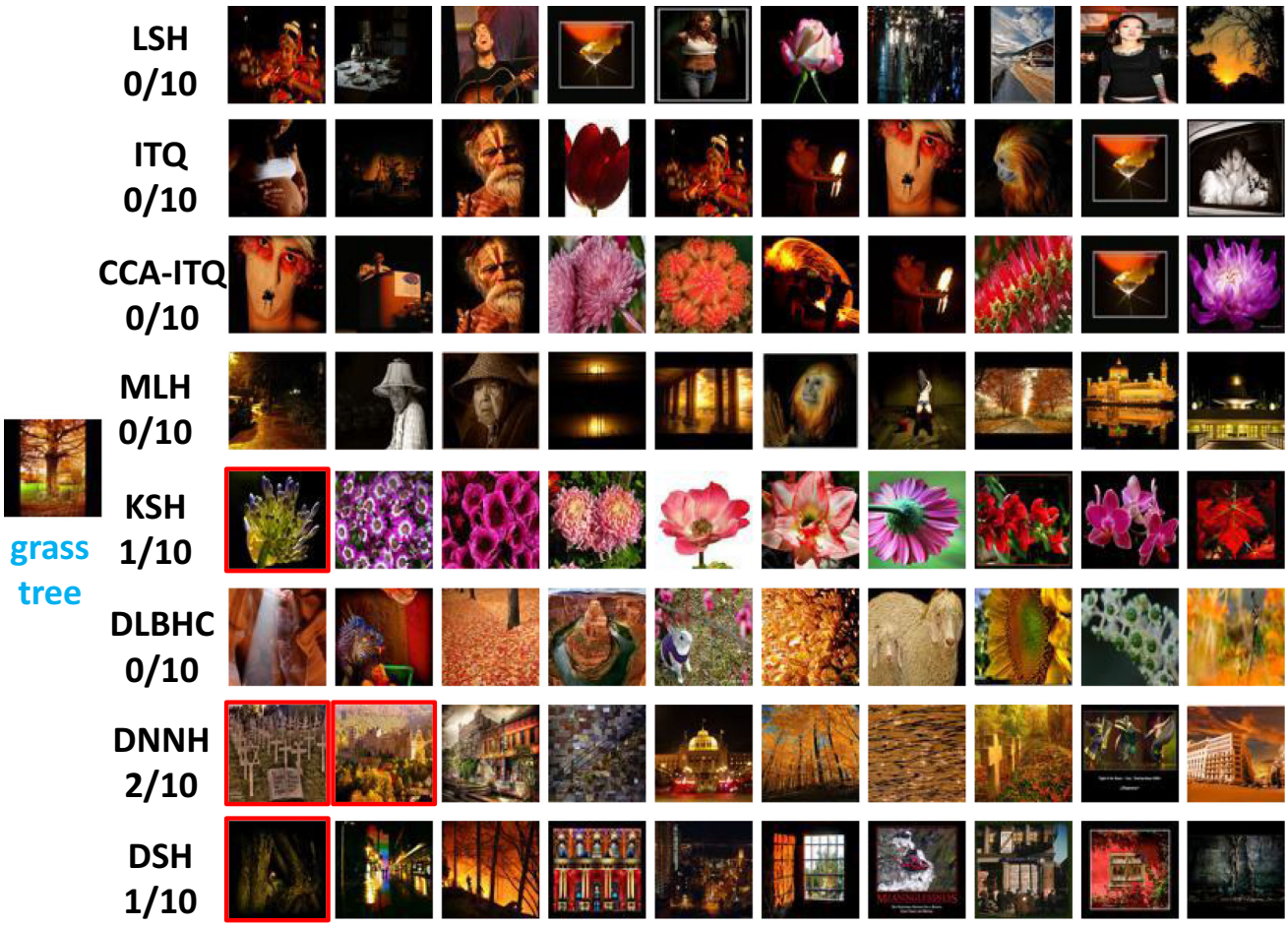

Fig. 10 A failed retrieval case on NUS-WIDE, only the top-10 feedbacks are shown due to space limitation. Results are obtained with 48-bit binary codes. Images with red boxes are true matches (Color figure online)

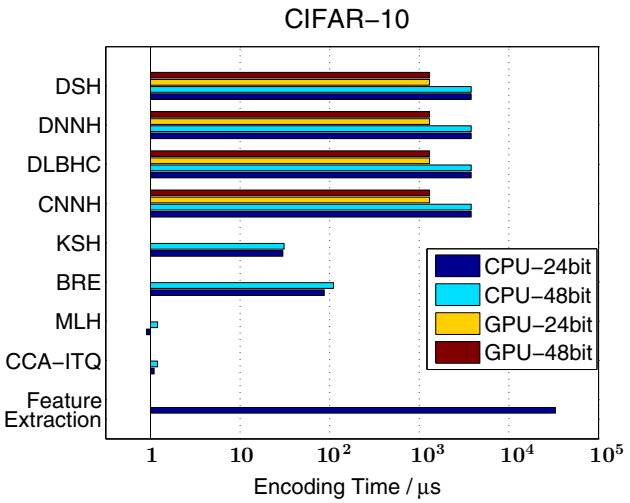

Fig. 11 Time cost to encode one new-coming image (microseconds) on CIFAR-10

The mAP results of the compared methods are listed in Table 6. We make four observations from these results. **First**, all compared data-dependent methods (ITQ, CCA-ITQ, DNNH, and DSH) perform better than LSH, suggesting that even the hash functions learned on an independent dataset are better than random projections. **Second**, all data-dependent methods performs similarly except for DNNH. A possible explanation is that DNNH does not use the tricks in Hermans et al. (2017) for training triplet loss, and such results again suggest that triplet loss is difficult to train on such large-scale datasets. **Third**, compared to Table 4, even though the CNN features are not specifically designed for the evaluated datasets, using such features can improve the performance of all non-deep methods (LSH, ITQ, and CCA-ITQ), again validating the advantage of deep features. **Fourth**, the performance drops significantly on CIFAR-10 (compared to DNNH & DSH in Table 4 and LSH & CCA-ITQ in Table 5), while the performance on NUS-WIDE only drops slightly (compared to DNNH and DSH in Table 4). Such a contrast could be attributed to the fact that images in NUS-WIDE are more similar to ImageNet images (both are high-resolution natural images), while images in CIFAR-10 are very different from ImageNet images (images in CIFAR-10 have much lower resolution compared to ImageNet images, i.e. 32×32

Table 6 Cross-dataset retrieval performance (mAP) of 4 existing methods and our proposed DSH method on CIFAR-10 and NUS-WIDE

Method	CIFAR-10	NUS-WIDE
LSH	0.142	0.379
ITQ	0.214	0.552
CCA-ITQ	0.205	0.559
DNNH	0.153	0.513
DSH	0.211	0.551

compared to 256×256). As a result, features learned on ImageNet can more easily transfer to NUS-WIDE and can hardly extract useful information from images on CIFAR-10.

6 Conclusions

In this paper we address the problem of large-scale content-based image retrieval. Considering the low time and memory costs of binary codes in retrieval tasks, we propose a Deep Supervised Hashing (DSH) method which jointly learns the image representation and hash functions in an end-to-end manner. Experiments on three challenging datasets validate the superiority of our proposed method. We attribute the promising retrieval performance of DSH to three aspects: **First**, the coupling of non-linear feature learning and hash coding for extracting task-specific image representations; **Second**, the proposed regularizer for reducing the discrepancy between the real-valued network output space and the desired Hamming space; **Third**, the online generated dense pairwise supervision for well describing the desired Hamming space. In terms of efficiency, experiments have shown that the proposed method encodes new-coming images even faster than conventional hashing methods. By combining our framework with more complex network structure, which could be done easily, the performance of our method is significantly improved. In addition, exploratory study of "network ensembles" in this work has proven it a promising way that is worth our future investigation to further boost retrieval performance.

Although we propose a regularizer to reduce the discrepancy between the network output space and the desired Hamming space, some outputs still fall around the threshold (i.e. 0) as shown in Fig. 4, which is likely to degrade the performance of the learned models. Inspired by recent works that constrain the weights and activations in the neural network to be binary (Rastegari et al. 2016; Soudry et al. 2014), we intend to explore directly optimizing the loss function in Hamming space in our future work.

Acknowledgements This work is partially supported by 973 Program under Contract No. 2015CB351802, Natural Science Foundation of China under Contracts Nos. 61390511, 61772500, Frontier Science Key Research Project CAS No. QYZDJ-SSW-JSC009, and Youth Innovation Promotion Association CAS No. 2015085.

References

Cao, Z., Long, M., Wang, J., & Yu, P. S. (2017). Hashnet: Deep learning to hash by continuation. In *The IEEE international conference on computer vision*, pp. 5608–5617.

Chua, T., Tang, J., Hong, R., Li, H., Luo, Z., & Zheng, Y. (2009). NUS-WIDE: A real-world web image database from National University of Singapore. In *Proceedings of the ACM international conference on image and video retrieval*, pp. 48:1–48:9.

Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., et al. (2014). Large-scale object classification using label relation graphs. In *European Conference on Computer Vision*, pp. 48–64.

Erin Liong, V., Lu, J., Wang, G., Moulin, P., & Zhou, J. (2015). Deep hashing for compact binary codes learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2475–2483.

Gionis, A., Indyk, P., & Motwani, R. (1999). Similarity search in high dimensions via hashing. In *Proceedings of 25th international conference on very large data bases*, pp. 518–529.

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256.

Gong, Y., & Lazebnik, S. (2011). Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 817–824.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Computer Society conference on computer vision and pattern recognition*, pp. 1735–1742.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778.

Hermans, A., Beyer, L., & Leibe, B. (2017). In defense of the triplet loss for person re-identification. ArXiv preprint arXiv:1703.07737.

Jégou, H., Douze, M., & Schmid, C. (2011). Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(1), 117–128.

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R. B., et al. (2014). Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on multimedia*, pp. 675–678.

Jiang, Q. Y., & Li, W. J. (2017). Deep cross-modal hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3270–3278.

Kang, W. C., Li, W. J., & Zhou, Z. H. (2016). Column sampling based discrete supervised hashing. In *Proceedings of the thirtieth AAAI conference on artificial intelligence*, pp. 1230–1236.

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105.

Kulis, B., & Darrell, T. (2009). Learning to hash with binary reconstructive embeddings. In *Advances in neural information processing systems*, pp. 1042–1050.

Lai, H., Pan, Y., Liu, Y., & Yan, S. (2015). Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3270–3278.

Li, W., Wang, S., & Kang, W. (2016). Feature learning based deep supervised hashing with pairwise labels. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence*, pp. 1711–1717.

Lin, K., Yang, H., Hsiao, J., & Chen, C. (2015a). Deep learning of binary hash codes for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 27–35.

Lin, T., RoyChowdhury, A., & Maji, S. (2015b). Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 1449–1457.

Liu, H., Wang, R., Shan, S., & Chen, X. (2016). Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2064–2072.

Liu, L., Shen, F., Shen, Y., Liu, X., & Shao, L. (2017). Deep sketch hashing: Fast free-hand sketch-based image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2298–2307.

Liu, W., Mu, C., Kumar, S., & Chang, S. (2014). Discrete graph hashing. In *Advances in neural information processing systems*, pp. 3419–3427.

Liu, W., Wang, J., Ji, R., Jiang, Y., & Chang, S. (2012). Supervised hashing with kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2074–2081.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning*, pp. 807–814.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*.

Norouzi, M., & Fleet, D. J. (2011). Minimal loss hashing for compact binary codes. In *Proceedings of the 28th international conference on machine learning*, pp. 353–360.

Norouzi, M., Fleet, D. J., & Salakhutdinov, R. (2012). Hamming distance metric learning. In *Advances in neural information processing systems*, pp. 1061–1069.

Oliva, A., & Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, *42*(3), 145–175.

Rastegari, M., Farhadi, A., & Forsyth, D. (2012). Attribute discovery via predictable discriminative binary codes. In *European Conference on Computer Vision*, pp. 876–889.

Rastegari, M., Ordonez, V., Redmon, J., & Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pp. 525–542.

Shen, F., Shen, C., Liu, W., & Shen, H. T. (2015). Supervised discrete hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 37–45.

Shen, L., Lin, Z., & Huang, Q. (2016). Relay backpropagation for effective learning of deep convolutional neural networks. In *European Conference on Computer Vision*, pp. 467–482.

Soudry, D., Hubara, I., & Meir, R. (2014). Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *Advances in neural information processing systems*, pp. 963–971.

Sun, Y., Chen, Y., Wang, X., Tang, X. (2014). Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pp. 1988–1996.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9.

Szegedy, C., Toshev, A., & Erhan, D. (2013). Deep neural networks for object detection. In *Advances in neural information processing systems*, pp. 2553–2561.

Wang, J., Kumar, S., & Chang, S. (2012). Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *34*(12), 2393–2406.

Wang, J., Zhang, T., Song, J., Sebe, N., & Shen, H. T. (2017). A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*, 769–790.

Wang, X., Shi, Y., & Kitani, K. M. (2016). Deep supervised hashing with triplet labels. In *Asian Conference on Computer Vision*, pp. 70–84.

Weiss, Y., Torralba, A., & Fergus, R. (2008). Spectral hashing. In *Advances in neural information processing systems*, pp. 1753–1760.

Xia, R., Pan, Y., Lai, H., Liu, C., & Yan, S. (2014). Supervised hashing for image retrieval via image representation learning. In *Proceedings of the twenty-eighth AAAI conference on artificial intelligence*, pp. 2156–2162.

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

Zhang, Z., Chen, Y., & Saligrama, V. (2016). Efficient training of very deep neural networks for supervised hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1487–1495.

Zhao, F., Huang, Y., Wang, L., & Tan, T. (2015). Deep semantic ranking based hashing for multi-label image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1556–1564.