Deep Supervised Hashing with Triplet Labels

Xiaofang Wang(✉), Yi Shi, and Kris M. Kitani

Carnegie Mellon University, Pittsburgh, PA 15213, USA
{xiaofan2,ys1}@andrew.cmu.edu, kkitani@cs.cmu.edu

Abstract. Hashing is one of the most popular and powerful approximate nearest neighbor search techniques for large-scale image retrieval. Most traditional hashing methods first represent images as off-the-shelf visual features and then produce hashing codes in a separate stage. However, off-the-shelf visual features may not be optimally compatible with the hash code learning procedure, which may result in sub-optimal hash codes. Recently, deep hashing methods have been proposed to simultaneously learn image features and hash codes using deep neural networks and have shown superior performance over traditional hashing methods. Most deep hashing methods are given supervised information in the form of pairwise labels or triplet labels. The current state-of-the-art deep hashing method DPSH [1], which is based on pairwise labels, performs image feature learning and hash code learning simultaneously by maximizing the likelihood of pairwise similarities. Inspired by DPSH [1], we propose a triplet label based deep hashing method which aims to maximize the likelihood of the given triplet labels. Experimental results show that our method outperforms all the baselines on CIFAR-10 and NUS-WIDE datasets, including the state-of-the-art method DPSH [1] and all the previous triplet label based deep hashing methods.

1 Introduction

With the rapid growth of image data on the Internet, much attention has been devoted to approximate nearest neighbor (ANN) search. Hashing is one of the most popular and powerful techniques for ANN search due to its computational and storage efficiencies. Hashing aims to map high dimensional image features into compact hash codes or binary codes so that the Hamming distance between hash codes approximates the Euclidean distance between image features.

Many hashing methods have been proposed and they can be categorized into data-independent and data-dependent methods. Compared with the data-dependent methods, data-independent methods need longer codes to achieve satisfactory performance [2]. Data-dependent methods can be further categorized into unsupervised and supervised methods. Compared to unsupervised methods, supervised methods usually can achieve competitive performance with fewer bits due to the help of supervised information, which is advantageous for search speed and storage efficiency [3].

Most existing hashing methods first represent images as off-the-shelf visual features such as GIST [4], SIFT [5] and hash code learning procedure is independent of the features of images. However, off-the-shelf visual features may not

S.-H. Lai et al. (Eds.): ACCV 2016, Part I, LNCS 10111, pp. 70–84, 2017.
DOI: 10.1007/978-3-319-54181-5_5

be optimally compatible with hash code learning procedure. In other words, the similarity between two images may not be optimally preserved by the visual features and thus the learned hash codes are sub-optimal [3]. Therefore, those hashing methods may not be able to achieve satisfactory performance in practice. To address the drawbacks of hashing methods that rely on off-the-shelf visual features, feature learning based deep hashing methods [1,3,6,7] have been proposed to simultaneously learn image feature and hash codes with deep neural networks and have demonstrated superior performance over traditional hashing methods. Most proposed deep hashing methods fall into the category of supervised hashing methods. Supervised information is given in the form of pairwise labels or triplet labels, a special case of ranking labels.

DPSH [1] is the current state-of-the-art deep hashing method, which is supervised by pairwise labels. Similar to LFH [7], DPSH aims to maximize the likelihood of the pairwise similarities, which is modeled as a function of the Hamming distance between the corresponding data points.

We argue that triplet labels inherently contain richer information than pairwise labels. Each triplet label can be naturally decomposed into two pairwise labels. Whereas, a triplet label can be constructed from two pairwise labels only when the same query image is used in a positive pairwise label and a negative pairwise label simultaneously. A triplet label ensures that in the learned hash code space, the query image is close to the positive image and far from the negative image simultaneously. However, a pairwise label can only ensure that one constraint is observed. Triplet labels explicitly provide a notion of relative similarities between images while pairwise labels can only encode that implicitly.

Therefore, we propose a triplet label based deep hashing method, which performs image feature learning and hash code learning simultaneously by maximizing the likelihood of the given triplet labels. As shown in Fig. 1, the proposed model has three key components: (1) image feature learning component: a deep neural network to learn visual features from images, (2) hash code learning component: one fully connected layer to learn hash codes from image features and (3) loss function component: a loss function to measure how well the given triplet labels are satisfied by the learned hash codes by computing the likelihood of the given triplet labels. Extensive experiments on standard benchmark datasets such as CIFAR-10 and NUS-WIDE show that our proposed deep hashing method outperforms all the baselines, including the state-of-the-art method DPSH [1] and all the previous triplet label based deep hashing methods.

Contributions: (1) We propose a novel triplet label based deep hashing method to simultaneously perform image feature and hash code learning in an end-to-end manner; (2) We present a novel formulation of the likelihood of the given triplet labels to evaluate the quality of learned hash codes; (3) We provide ablative analysis of our loss function to help understand how each term contributes to performance; (4) We obtain state-of-the-art performance on benchmark datasets.

2 Related Work

Hashing methods can be categorized into data-independent and data-dependent methods, based on whether they are independent of training data. Representative data-independent methods include Locality Sensitive Hashing (LSH) [8] and Shift-Invariant Kernels Hashing (SIKH) [9]. Data-independent methods generally need longer hash codes for satisfactory performance, compared to data-dependent methods. Data-dependent methods can be further divided into unsupervised and supervised methods, based on whether supervised information is provided or not.

Unsupervised hashing methods only utilize the training data points to learn hash functions, without using any supervised information. Notable examples for unsupervised hashing methods include Spectral Hashing (SH) [10], Binary Reconstructive Embedding (BRE) [11], Iterative Quantization (ITQ) [2], Isotropic Hashing (IsoHash) [12], graph-based hashing methods [13–15] and two deep hashing methods: Semantic Hashing [16] and the hashing method proposed in [17].

Supervised hashing methods leverage labeled data to learn hash codes. Typically, the supervised information are provided in one of three forms: point-wise labels, pairwise labels or ranking labels [1]. Representative point-wise label based methods include CCA-ITQ [2] and the deep hashing method proposed in [18]. Representative pairwise label based hashing methods include Minimal Loss Hashing (MLH) [19], Supervised Hashing with Kernels (KSH) [20], Latent Factor Hashing (LFH) [21], Fash Supervised Hashing (FASTH) [22] and two deep hashing methods: CNNH [23] and DPSH [1]. Representative ranking label based hashing methods include Ranking-based Supervised Hashing (RSH) [24], Column Generation Hashing (CGHASH) [25] and the deep hashing methods proposed in [3,6,7,17]. One special case of ranking labels is triplet labels.

Most existing supervised hashing methods represent images as off-the-shelf visual features and perform hash code learning independent of visual features, including some deep hashing methods [16,17]. However, off-the-shelf features may not be optimally compatible with hash code learning procedure and thus results in sub-optimal hash codes.

CNNH [23], supervised by triplet labels, is the first proposed deep hashing method without using off-the-shelf features. However, CNNH cannot learn image features and hash codes simultaneously and still has limitations. This has been verified by the authors of CNNH themselves in a follow-up work [3]. Other ranking label or triplet label based deep hashing methods include Network in Network Hashing (NINH) [3], Deep Semantic Ranking based Hashing (DSRH) [6], Deep Regularized Similarity Comparison Hashing (DRSCH) [7] and Deep Similarity Comparison Hashing (DSCH) [7]. While these methods can simultaneously perform image feature learning and hash code learning given supervision of triplet labels, we present a novel formulation of the likelihood of the given triplet labels to evaluate the quality of learned hash codes.

Deep Pairwise-Supervised Hashing (DPSH) [1] is the first proposed deep hashing method to simultaneously perform image feature learning and hash code learning with pairwise labels and achieves highest performance compared to

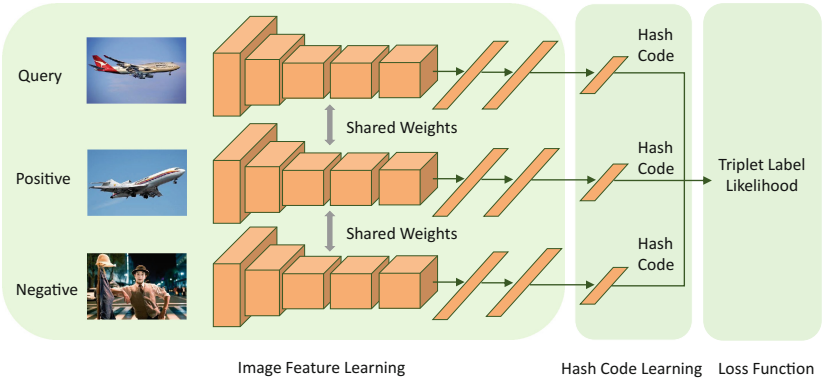


Fig. 1. Overview of the proposed end-to-end deep hashing method. (Color figure online)

other deep hashing methods. Our method is supervised by triplet labels because triplet labels inherently contain richer information than pairwise labels.

3 Approach

We first give the formal definition of our problem and then introduce our proposed end-to-end method which simultaneously performs image feature learning and hash code learning from triplet labels.

3.1 Problem Definition

Given N training images $\mathcal{I} = \{I_1, \ldots, I_N\}$ and M triplet labels $\mathcal{T} = \{(q_1, p_1, n_1), \ldots, (q_M, p_M, n_M)\}$ where the triplet of image indices (q_m, p_m, n_m) denotes that the query image of index $q_m \in \{1 \ldots N\}$ is more similar to the positive image of index $p_m \in \{1 \ldots N\}$ than to the negative image of index $n_m \in \{1 \ldots N\}$. One possible way of generating triplet labels is by selecting two images from the same semantic category (I_{q_m} and I_{p_m}) and selecting the negative (I_{n_m}) from a different semantic category.

Our goal is to learn a hash code $\mathbf{b}_n$ for each image I_n, where $\mathbf{b} \in \{+1, -1\}^L$ and L is the length of hash codes. The hash codes $\mathcal{B} = \{\mathbf{b}_n\}_{n=1}^N$ should satisfy all the triplet labels $\mathcal{T}$ as much as possible in the Hamming space. More specifically, $\text{dist}_H(\mathbf{b}_{q_m}, \mathbf{b}_{p_m})$ should be smaller than $\text{dist}_H(\mathbf{b}_{q_m}, \mathbf{b}_{n_m})$ as much as possible, where $\text{dist}_H(\cdot, \cdot)$ denotes the Hamming distance between two binary codes. Generally speaking, we aim to learn a hash function $h(\cdot)$ to map images to hash codes. We can write $h(\cdot)$ as $[h_1(\cdot), ..., h_L(\cdot)]$ and for image I_n, its hash code can be denoted as $\mathbf{b}_n = h(I_n) = [h_1(I_n), ..., h_L(I_n)]$.

3.2 Learning the Hash Function

Most previous hashing methods rely on off-the-shelf visual features, which may not be optimally compatible with the hash code learning procedure. Thus deep

hashing methods [1,3,6,7] are proposed to simultaneously learn image features and hash codes from images. We propose a novel deep hashing method utilizing triplet labels, which simultaneously performs image feature learning and hash code learning in an end-to-end manner. As shown in Fig. 1, our method consists of three key components: (1) an image feature learning component, (2) a hash code learning component and (3) a loss function component.

Image Feature Learning. This component is designed to employ a Convolutional neural network to learn visual features from images. We adopt the CNN-F network architecture [26] for this component. CNN-F has eight layers, where the last layer is designed to learn the probability distribution over category labels. So only the first 7 layers of CNN-F are used in this component. Other networks like AlexNet [27], residual network [28] can also be used in this component.

Hash Code Learning. This component is designed to learn hash codes of images. We use one fully connected layer for this component and we want this layer to output hash codes of images. In particular, the number of neurons of this layer equals the length of targeted hash codes. Multiple fully connected layers or other architectures like the divide-and-encode module proposed by [3] can also be applied here. We do not focus on this in this work and leave this for future study.

Loss Function. This component measures how well the given triple labels are satisfied by the learned hash codes by computing the likelihood of the given triplet labels. Inspired by the likelihood of the pairwise similarities proposed in LFH [21], we present our formulation of the likelihood for a given triplet label. We call this the *triplet label likelihood* throughout the text.

Let Θ_{ij} denote half of the inner product between two hash codes $\mathbf{b}_i, \mathbf{b}_j \in \{+1, -1\}^L$:

$$\Theta_{ij} = \frac{1}{2}\mathbf{b}_i^T \mathbf{b}_j \tag{1}$$

Then the triplet label likelihood is formulated as:

$$p(\mathcal{T} \mid \mathcal{B}) = \prod_{m=1}^{M} p((q_m, p_m, n_m) \mid \mathcal{B}) \tag{2}$$

with

$$p((q_m, p_m, n_m) \mid \mathcal{B}) = \sigma(\Theta_{q_m p_m} - \Theta_{q_m n_m} - \alpha) \tag{3}$$

where $\sigma(x)$ is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$, α is the margin, a positive hyper-parameter and $\mathcal{B}$ is the set of all hash codes.

We now show how maximizing the triplet label likelihood matches the goal to preserve the relative similarity between the query image, positive image and

negative image. We first prove the following relationship between the Hamming distance between two binary codes and their inner product:

$$\text{dist}_H(\mathbf{b}_i, \mathbf{b}_j) = \frac{1}{2}(L - 2\Theta_{ij}) \tag{4}$$

According to Eq. 4, we can have

$$\text{dist}_H(\mathbf{b}_{q_m}, \mathbf{b}_{p_m}) - \text{dist}_H(\mathbf{b}_{q_m}, \mathbf{b}_{n_m}) = -(\Theta_{q_m p_m} - \Theta_{q_m n_m}) \tag{5}$$

According to Eq. 3, we know that the larger $p((q_m, p_m, n_m) \mid \mathcal{B})$ is, the larger $(\Theta_{q_m p_m} - \Theta_{q_m n_m} - \alpha)$ will be. Since α is a constant number here, the larger $(\Theta_{q_m p_m} - \Theta_{q_m n_m} - \alpha)$ is, the smaller $(\text{dist}_H(\mathbf{b}_{q_m}, \mathbf{b}_{p_m}) - \text{dist}_H(\mathbf{b}_{q_m}, \mathbf{b}_{n_m}))$ will be. Thus, by maximizing the triplet label likelihood $p(\mathcal{T} \mid \mathcal{B})$, we can enforce the Hamming distance between the query and the positive image to be smaller than that between the query and the negative image. The margin α here can regularize the distance gap between $\text{dist}_H(\mathbf{b}_{q_m}, \mathbf{b}_{p_m})$ and $\text{dist}_H(\mathbf{b}_{q_m}, \mathbf{b}_{n_m})$. The margin α can also help speed up training our model as explained later in this Section and verified in our experiments in Sect. 4.4.

Now we define our loss function as the negative log triplet label likelihood as follows,

$$\begin{aligned} L &= -\log p(\mathcal{T} \mid \mathcal{B}) \\ &= -\sum_{m=1}^{M} \log p((q_m, p_m, n_m) \mid \mathcal{B}) \end{aligned} \tag{6}$$

Plug Eq. 3 into the above equation, we can drive that

$$L = -\sum_{m=1}^{M} (\Theta_{q_m p_m} - \Theta_{q_m n_m} - \alpha - \log(1 + e^{\Theta_{q_m p_m} - \Theta_{q_m n_m} - \alpha})) \tag{7}$$

Minimizing the loss defined in (7) is an intractable discrete optimization problem [1]. One can choose to relax $\{\mathbf{b}_n\}$ from discrete to continuous, *i.e.*, relax $\{\mathbf{b}_n\}$ to $\{\mathbf{u}_n\}$, where $\mathbf{u}_n \in \mathbb{R}^{L \times 1}$ and then minimize the loss. This is the strategy employed by LFH [21], but this may be harmful to the performance due to the relaxation error [29]. In the context of hashing, the relaxation error is actually the quantization error. Although optimal real vectors $\{\mathbf{u}_n\}$ are learned, we still need to quantize them to binary codes $\{\mathbf{b}_n\}$. This process induces quantization error. To reduce it, we propose to also consider the quantization error when solving the relaxed problem.

Concretely, we relax binary codes $\{\mathbf{b}_n\}$ to real vectors $\{\mathbf{u}_n\}$ and re-define Θ_{ij} as

$$\Theta_{ij} = \frac{1}{2}\mathbf{u}_i^T \mathbf{u}_j \tag{8}$$

and our loss function becomes

$$L = -\sum_{m=1}^{M}(\Theta_{q_m p_m} - \Theta_{q_m n_m} - \alpha - \log(1 + e^{\Theta_{q_m p_m} - \Theta_{q_m n_m} - \alpha})) + \lambda \sum_{n=1}^{N} ||\mathbf{b}_n - \mathbf{u}_n||_2^2 \tag{9}$$

where λ is a hyper-parameter to balance the negative log triplet likelihood and the quantization error and $\mathbf{b}_n = sgn(\mathbf{u}_n)$, where $sgn(\cdot)$ is the sign function and $sgn(\mathbf{u}_n^{(k)})$ equals to 1 if $\mathbf{u}_n^{(k)} > 0$ and -1, otherwise. The quantization error term $\sum_{n=1}^{N} ||\mathbf{b}_n - \mathbf{u}_n||_2^2$ is also adopted as a regularization term in DPSH [1]. However, they do not interpret it as quantization error.

Model Learning. The three key components of our method can be integrated into a Siamese-triplet network as shown in Fig. 1, which takes triplets of images as input and output hash codes of images. The network consists of three sub-networks with exactly the same architecture and shared weights. In our experiments, the sub-network is a fully connected layer on top of the first seven layers of CNN-F [26]. We train our network by minimizing the loss function:

$$L(\theta) = -\sum_{m=1}^{M}(\Theta_{q_m p_m} - \Theta_{q_m n_m} - \alpha - \log(1 + e^{\Theta_{q_m p_m} - \Theta_{q_m n_m} - \alpha})) + \lambda \sum_{n=1}^{N} ||\mathbf{b}_n - \mathbf{u}_n||_2^2 \tag{10}$$

where θ denotes all the parameters of the sub-network, $\mathbf{u}_n$ is the output of the sub-network with n^{th} training image and $\mathbf{b}_n = sgn(\mathbf{u}_n)$. We can see that L is differentiable with respect to $\mathbf{u}_n$. Thus the back-propagation algorithm can be applied here to minimize the loss function.

Once training is completed, we can apply our model to generate hash codes for new images. For a new image I, we pass it into the trained sub-network and take the output of the last layer $\mathbf{u}$. Then the hash code $\mathbf{b}$ of image I is $\mathbf{b} = sgn(\mathbf{u})$.

Impact of Margin α. We argue that a positive margin α can help speed up the training process. We now analyze this theoretically by looking at the derivative of the loss function. In particular, for n^{th} image, we compute the derivative of

the loss function L with respect to $\mathbf{u}_n$ as follows:

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{u}_n} = &-\frac{1}{2}\sum_{m:(n,p_m,n_m)\in\mathcal{T}}(1-\sigma(\Theta_{q_m p_m}-\Theta_{q_m n_m}-\alpha))(\mathbf{u}_{p_m}-\mathbf{u}_{n_m})\\ &-\frac{1}{2}\sum_{m:(q_m,n,n_m)\in\mathcal{T}}(1-\sigma(\Theta_{q_m p_m}-\Theta_{q_m n_m}-\alpha))\mathbf{u}_{q_m}\\ &+\frac{1}{2}\sum_{m:(q_m,p_m,n)\in\mathcal{T}}(1-\sigma(\Theta_{q_m p_m}-\Theta_{q_m n_m}-\alpha))\mathbf{u}_{q_m}\\ &+2\lambda(\mathbf{u}_n-\mathbf{b}_n)\end{aligned} \tag{11}$$

where $\sigma(x)$ is the sigmoid function $\sigma(x)=\frac{1}{1+e^{-x}}$ and $\mathcal{T}$ is the set of triplet labels. In the derivative shown above, we observe this term $(1-\sigma(\Theta_{q_m p_m}-\Theta_{q_m n_m}-\alpha))$. We know that $\sigma(x)$ saturates very quickly, *i.e.*, being very close to 1, as x increases. If $\alpha=0$, when $(\Theta_{q_m p_m}-\Theta_{q_m n_m})$ becomes positive, the term $(1-\sigma(\Theta_{q_m p_m}-\Theta_{q_m n_m}-\alpha))$ will be very close to 0. This will make the magnitude of the derivative very small and further make the model hard to train. A positive margin α adds a negative offset on $(\Theta_{q_m p_m}-\Theta_{q_m n_m})$ and can prevent $(1-\sigma(\Theta_{q_m p_m}-\Theta_{q_m n_m}-\alpha))$ from being very small. Further this makes the model easier to train and helps speed up the training process. We give experiment results to verify this in Sect. 4.4.

4 Experiment

4.1 Datasets and Evaluation Protocol

We conduct experiments on two widely used benchmark datasets, CIFAR-10 [30] and NUS-WIDE [31]. The CIFAR-10 [30] dataset contains 60,000 color images of size 32×32, which can be divided into 10 categories and 6,000 images for each category. Each image is only associated with one category. The NUS-WIDE [31] dataset contains nearly 27,000 color images from the web. Different from CIFAR-10, NUS-WIDE is a multi-label dataset. Each image is annotated with one or multiple labels in 81 semantic concepts. Following the setting in [1,3,7,23], we only consider images annotated with 21 most frequent labels. For each of the 21 labels, at least 5,000 images are annotated with the label. In addition, NUS-WIDE provides links to images for downloading and some links are now invalid. This causes some differences between the image set used by previous work and our work. In total, we use 161,463 images from the NUS-WIDE dataset.

We employ mean average precision (MAP) to evaluate the performance of our method and baselines similar to most previous work [1,3,7,23]. Two images in CIFAR-10 are considered similar if they belong to the same category. Two images in NUS-WIDE are considered similar if they share at least one label.

4.2 Baselines and Setting

Following [1], we consider the following baselines:

1. Traditional unsupervised hashing methods using hand-crafted features, including SH [10], ITQ [2].
2. Traditional supervised hashing methods using hand-crafted features, including SPLH [32], KSH [20], FastH [22], LFH [21] and SDH [33].
3. The above traditional hashing methods using features extracted by CNN-F network [26] pre-trained on ImageNet.
4. Pairwise label based deep hashing methods: CNNH [23] and DPSH [1].
5. Triplet label based deep hashing methods: NINH [3], DSRH [6], DSCH [7] and DRSCH [7].

When using hand-crafted features, we use a 512-dimensional GIST descriptor [4] to represent CIFAR-10 images. For NUS-WIDE images, we represent them by a 1134-dimensional feature vector, which is the concatenation of a 64-D color histogram, a 144-D color correlogram, a 73-D edge direction histogram, a 128-D wavelet texture, a 225-D block-wise color moments and a 500-D BoW representation based on SIFT descriptors.

Following [1,6], we initialize the first seven layers of our network with the CNN-F network [26] pre-trained on ImageNet. In addition, the hyper-parameter α is set to half of the length of hash codes, *e.g.*, 16 for 32-bit hash codes and the hyper-parameter λ is set to 100 unless otherwise stated.

We compare our method to most baselines under the following experimental setting. Following [1,3,23], in CIFAR-10, 100 images per category, *i.e.*, in total 1,000 images, are randomly sampled as query images. The remaining images are used as database images. For unsupervised hashing methods, all the database images are used as training images. For supervised hashing methods, 500 database images per category, *i.e.*, in total 5,000 images, are randomly sampled as training images. In NUS-WIDE, 100 images per label, *i.e.*, in total 2,100 images, are randomly sampled as query images. Likewise, the remaining images are used as database images. For unsupervised hashing methods, all the database images are used as training images. For supervised hashing methods, 500 database images per label, *i.e.*, in total 10,500 images, are randomly sampled as training images. Since NUS-WIDE contains a huge number of images, when computing MAP for NUS-WIDE, only the top 5,000 returned neighbors are considered.

We also compare our method to DSRH [6], DSCH [7], DRSCH [7] and DPSH [1] under a different experimental setting. In CIFAR-10, 1,000 images per category, *i.e.*, in total 10,000 images, are randomly sampled as query images. The remaining images are used as database images and all the database images are used as training images. In NUS-WIDE, 100 images per label, *i.e.*, in total 2,100 images, are randomly sampled as query images. The remaining images are used as database images and still, all the database images are used as training images. Under this setting, when computing MAP for NUS-WIDE, we only consider the top 50,000 returned neighbors.

4.3 Performance Evaluation

Comparison to Traditional Hashing Methods using Hand-crafted Features. As shown in Table 1, we can see that on both datasets, our method

Table 1. Mean Average Precision (MAP) under the first experimental setting. The MAP for NUS-WIDE is computed based on the top 5,000 returned neighbors. The best performance is shown in boldface. DPSH* denotes the performance we obtain by running the code provided by the authors of DPSH in our experiments.

Method	CIFAR-10				Method	NUS-WIDE			
	12 bits	24 bits	32 bits	48 bits		12 bits	24 bits	32 bits	48 bits
Ours	0.710	**0.750**	**0.765**	**0.774**	Ours	**0.773**	**0.808**	**0.812**	**0.824**
DPSH	**0.713**	0.727	0.744	0.757	DPSH*	0.752	0.790	0.794	0.812
NINH	0.552	0.566	0.558	0.581	NINH	0.674	0.697	0.713	0.715
CNNH	0.439	0.511	0.509	0.522	CNNH	0.611	0.618	0.625	0.608
FastH	0.305	0.349	0.369	0.384	FastH	0.621	0.650	0.665	0.687
SDH	0.285	0.329	0.341	0.356	SDH	0.568	0.600	0.608	0.637
KSH	0.303	0.337	0.346	0.356	KSH	0.556	0.572	0.581	0.588
LFH	0.176	0.231	0.211	0.253	LFH	0.571	0.568	0.568	0.585
SPLH	0.171	0.173	0.178	0.184	SPLH	0.568	0.589	0.597	0.601
ITQ	0.162	0.169	0.172	0.175	ITQ	0.452	0.468	0.472	0.477
SH	0.127	0.128	0.126	0.129	SH	0.454	0.406	0.405	0.400

outperforms previous hashing methods using hand-crafted features significantly. In Table 1, the results of NINH, CNNH, KSH and ITQ are from [3,23] and the results of other methods except our method are from [1]. This is reasonable as we use the same experimental setting and evaluation protocol.

Comparison to Traditional Hashing Methods using Deep Features. When we train our model, we initialize the first 7 layers of our network with CNN-F network [26] pre-trained on ImageNet. Thus one may argue that the boost in performance comes from that network instead of our method. To further validate our method, we compare our method to traditional hashing methods using deep features extracted by CNN-F network. As shown in Table 2, we can see that our method can significantly outperform traditional methods on CIFAR-10 and obtain comparable performance with the best performing traditional methods on NUS-WIDE. The results in Table 2 are copied from [1], which is reasonable as we used the same experimental setting and evaluation protocol.

Comparison to Deep Hashing Methods. Now we compare our method to other deep hashing methods. In particular, we compare our method to CNNH, NINH and DPSH under the first experimental setting in Table 1 and DSRH, DSCH, DRSCH and DPSH under the second experimental setting in Table 3. The results of DSRH, DSCH and DRSCH are directly from [7]. We can see that our method significantly outperforms all previous triplet label based deep hashing methods, including NINH, DSRH, DSCH and DRSCH.

Table 2. Mean Average Precision (MAP) under the first experimental setting. The MAP for NUS-WIDE is computed based on the top 5,000 returned neighbors. The best performance is shown in boldface.

Method	CIFAR-10				NUS-WIDE			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits
Ours	**0.710**	**0.750**	**0.765**	**0.774**	0.773	**0.808**	0.812	0.824
FastH + CNN	0.553	0.607	0.619	0.636	0.779	0.807	**0.816**	**0.825**
SDH + CNN	0.478	0.557	0.584	0.592	**0.780**	0.804	0.815	0.824
KSH + CNN	0.488	0.539	0.548	0.563	0.768	0.786	0.79	0.799
LFH + CNN	0.208	0.242	0.266	0.339	0.695	0.734	0.739	0.759
SPLH + CNN	0.299	0.33	0.335	0.33	0.753	0.775	0.783	0.786
ITQ + CNN	0.237	0.246	0.255	0.261	0.719	0.739	0.747	0.756
SH + CNN	0.183	0.164	0.161	0.161	0.621	0.616	0.615	0.612

Table 3. Mean Average Precision (MAP) under the second experimental setting. The MAP for NUS-WIDE is computed based on the top 50,000 returned neighbors. The best performance is shown in boldface. DPSH* denotes the performance we obtain by running the code provided by the authors of DPSH in our experiments.

Method	CIFAR-10				NUS-WIDE			
	16 bits	24 bits	32 bits	48 bits	16 bits	24 bits	32 bits	48 bits
Ours	**0.915**	**0.923**	**0.925**	**0.926**	**0.756**	**0.776**	**0.785**	**0.799**
DPSH	0.763	0.781	0.795	0.807	0.715	0.722	0.736	0.741
DRSCH	0.615	0.622	0.629	0.631	0.618	0.622	0.623	0.628
DSCH	0.609	0.613	0.617	0.62	0.592	0.597	0.611	0.609
DSRH	0.608	0.611	0.617	0.618	0.609	0.618	0.621	0.631
DPSH*	0.903	0.885	0.915	0.911	N/A			

We now compare our method to the current state-of-the-art method DPSH under the first experimental setting. As shown in Table 1, our method outperforms DPSH by about 2% on both CIFAR-10 and NUS-WIDE datasets. Note that on NUS-WIDE, we are comparing to DPSH* instead of DPSH. DPSH represents the performance reported in [1] and DPSH* represents the performance we obtain by running the code of DPSH provided by the authors of [1] on NUS-WIDE. We re-run their code on NUS-WIDE because the NUS-WIDE dataset does not provide the original images to download instead of the links to image, which results in some differences between the images used by them [1] and us.

As shown in Table 3, our method outperforms DPSH by more than 10% on CIFAR-10 and about 5% on NUS-WIDE under the second experimental setting. We also re-run the code of DPSH on CIFAR-10 under the same setting and

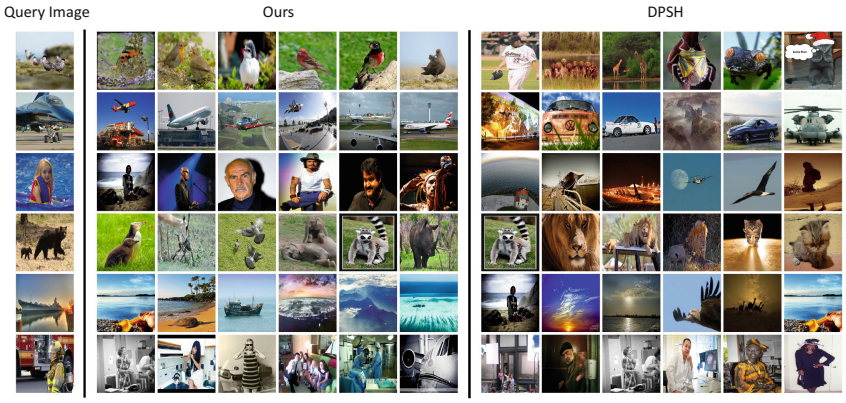


Fig. 2. Retrieval examples. Left: Query images. Middle: Top images retrieved by hash codes learnt by our method. Right: Top images retrieved by hash codes learnt by DPSH [1].

we can obtain much higher the performance then what was reported in [1].[1] The performance we obtain is denoted by DPSH* in Table 3 and we can see our method still outperforms DPSH* by about 1%. We also show some retrieval examples on NUS-WIDE with hash codes learned by our method and DPSH respectively in Fig. 2.

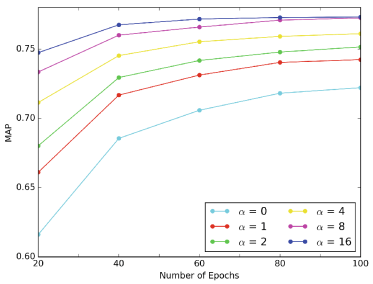
4.4 Ablation Studies

Impact of the hyper-parameter α. Figure 3a shows the effect of the margin α at 32 bits on CIFAR-10 dataset. We can see that within the same number of training epochs, we can obtain better performance with a larger margin. This verifies our previous analysis in Sect. 3.

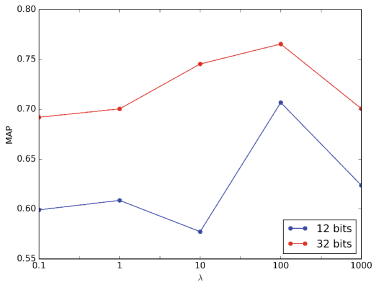
Impact of the hyper-parameter λ. Figure 3b shows the effect of the hyper-parameter λ in Eq. 9 at 12 bits and 32 bits on CIFAR-10 dataset. As one can see, for both 12 bits and 32 bits, there is a significant performance drop in terms of MAP when λ becomes very small (*e.g.*, 0.1) or very large (*e.g.*, 1000). This is reasonable since λ is designed to balance the negative log triplet likelihood and the quantization error. Setting λ to small or to large will lead to inbalance between these two terms.

Impact of the number of training images. We also study the impact of the number of training images on the performance. Figure 3c shows the performance

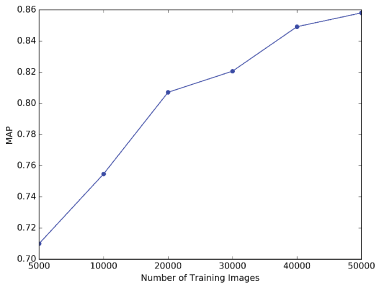
[1] We communicated with the authors of DPSH. The main difference between our experiments and their experiments is the step size and decay factor for learning rate change. They say that with our parameters, they can also get better results than what is reported in their paper [1].



(a) Impact of α



(b) Impact of λ



(c) Impact of number of training images

Fig. 3. Ablation studies.

of our method at 12 bits on CIFAR-10 using different number of training images. We can see that more training images will incur noticeable improvements.

5 Conclusion

In this paper, we have proposed a novel deep hashing method to simultaneously learn image features and hash codes given the supervision of triplet labels. Our method learns high quality hash codes by maximizing the likelihood of given triplet labels under learned hash codes. Extensive experiments on standard benchmark datasets show that our method outperforms all the baselines, including the state-of-the-art method DPSH [1] and all the previous triplet label based deep hashing methods.

Acknowledgement. This work was sponsored by DARPA under agreement number FA8750-14-2-0244. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

1. Li, W.J., Wang, S., Kang, W.C.: Feature learning based deep supervised hashing with pairwise labels. arXiv preprint arXiv:1511.03855 (2015)
2. Gong, Y., Lazebnik, S.: Iterative quantization: a procrustean approach to learning binary codes. In: 2011 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 817–824. IEEE (2011)
3. Lai, H., Pan, Y., Liu, Y., Yan, S.: 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 (2015)
4. Oliva, A., Torralba, A.: Modeling the shape of the scene: a holistic representation of the spatial envelope. Int. J. Comput. Vision **42**, 145–175 (2001)
5. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**, 91–110 (2004)
6. Zhao, F., Huang, Y., Wang, L., Tan, T.: 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 (2015)
7. Zhang, R., Lin, L., Zhang, R., Zuo, W., Zhang, L.: Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. IEEE Trans. Image Process. **24**, 4766–4779 (2015)
8. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: 47th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2006, pp. 459–468. IEEE (2006)
9. Raginsky, M., Lazebnik, S.: Locality-sensitive binary codes from shift-invariant kernels. In: Advances in Neural Information Processing Systems, pp. 1509–1517 (2009)
10. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances in Neural Information Processing Systems, pp. 1753–1760 (2009)
11. Kulis, B., Darrell, T.: Learning to hash with binary reconstructive embeddings. In: Advances in Neural Information Processing Systems, pp. 1042–1050 (2009)
12. Kong, W., Li, W.J.: Isotropic hashing. In: Advances in Neural Information Processing Systems, pp. 1646–1654 (2012)
13. Liu, W., Wang, J., Kumar, S., Chang, S.F.: Hashing with graphs. In: Proceedings of the 28th International Conference on Machine Learning (ICML 2011), pp. 1–8 (2011)
14. Liu, W., Mu, C., Kumar, S., Chang, S.F.: Discrete graph hashing. In: Advances in Neural Information Processing Systems, pp. 3419–3427 (2014)
15. Jiang, Q.Y., Li, W.J.: Scalable graph hashing with feature transformation. In: Proceedings of the International Joint Conference on Artificial Intelligence (2015)
16. Salakhutdinov, R., Hinton, G.: Semantic hashing. Int. J. Approximate Reasoning **50**, 969–978 (2009)
17. Erin Liong, V., Lu, J., Wang, G., Moulin, P., Zhou, J.: Deep hashing for compact binary codes learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2475–2483 (2015)
18. Lin, K., Yang, H.F., Hsiao, J.H., Chen, C.S.: 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 (2015)
19. Norouzi, M., Fleet, D.J.: Minimal loss hashing for compact binary codes. In: ICML, vol. 1, p. 2 (2011)

20. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2074–2081. IEEE (2012)
21. Zhang, P., Zhang, W., Li, W.J., Guo, M.: Supervised hashing with latent factor models. In: Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 173–182. ACM (2014)
22. Lin, G., Shen, C., Shi, Q., Hengel, A., Suter, D.: Fast supervised hashing with decision trees for high-dimensional data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1963–1970 (2014)
23. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: AAAI, vol. 1, p. 2 (2014)
24. Wang, J., Liu, W., Sun, A., Jiang, Y.G.: Learning hash codes with listwise supervision. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3032–3039 (2013)
25. Li, X., Lin, G., Shen, C., Van den Hengel, A., Dick, A.: Learning hash functions using column generation. In: Proceedings of The 30th International Conference on Machine Learning, pp. 142–150(2013)
26. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: delving deep into convolutional nets. arXiv preprint arXiv:1405.3531 (2014)
27. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 1097–1105 (2012)
28. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
29. Kang, W.C., Li, W.J., Zhou, Z.H.: Column sampling based discrete supervised hashing (2016)
30. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images (2009)
31. 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: Proceedings of the ACM International Conference on Image and Video Retrieval, p. 48. ACM (2009)
32. Wang, J., Kumar, S., Chang, S.F.: Sequential projection learning for hashing with compact codes. In: Proceedings of the 27th International Conference on Machine Learning (ICML 2010), pp. 1127–1134 (2010)
33. Shen, F., Shen, C., Liu, W., Tao Shen, H.: Supervised discrete hashing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 37–45 (2015)