Feature Learning Based Deep Supervised Hashing with Pairwise Labels

Wu-Jun Li, Sheng Wang and **Wang-Cheng Kang**
National Key Laboratory for Novel Software Technology
Department of Computer Science and Technology, Nanjing University, China
liwujun@nju.edu.cn, wangs@lamda.nju.edu.cn, kwc.oliver@gmail.com

Abstract

Recent years have witnessed wide application of hashing for large-scale image retrieval. However, most existing hashing methods are based on hand-crafted features which might not be optimally compatible with the hashing procedure. Recently, deep hashing methods have been proposed to perform simultaneous feature learning and hash-code learning with deep neural networks, which have shown better performance than traditional hashing methods with hand-crafted features. Most of these deep hashing methods are supervised whose supervised information is given with triplet labels. For another common application scenario with pairwise labels, there have not existed methods for simultaneous feature learning and hash-code learning. In this paper, we propose a novel deep hashing method, called deep pairwise-supervised hashing (DPSH), to perform simultaneous feature learning and hash-code learning for applications with pairwise labels. Experiments on real datasets show that our DPSH method can outperform other methods to achieve the state-of-the-art performance in image retrieval applications.

1 Introduction

With the explosive growing of data in real applications like image retrieval, approximate nearest neighbor (ANN) search [Andoni and Indyk, 2006] has become a hot research topic in recent years. Among existing ANN techniques, hashing has become one of the most popular and effective techniques due to its fast query speed and low memory cost [Kulis and Grauman, 2009; Gong and Lazebnik, 2011; Kong and Li, 2012; Liu *et al.*, 2012; Rastegari *et al.*, 2013; He *et al.*, 2013; Lin *et al.*, 2014; Shen *et al.*, 2015; Kang *et al.*, 2016].

Existing hashing methods can be divided into data-independent methods and data-dependent methods [Gong and Lazebnik, 2011; Kong and Li, 2012]. In data-independent methods, the hash function is typically randomly generated which is independent of any training data. The representative data-independent methods include locality-sensitive hashing (LSH) [Andoni and Indyk, 2006] and its variants. Data-dependent methods try to learn the hash function from some training data, which is also called learning to hash (L2H) methods [Kong and Li, 2012]. Compared with data-independent methods, L2H methods can achieve comparable or better accuracy with shorter hash codes. Hence, L2H methods have become more and more popular than data-independent methods in real applications.

The L2H methods can be further divided into two categories [Kong and Li, 2012; Kang *et al.*, 2016]: unsupervised methods and supervised methods. Unsupervised methods only utilize the feature (attribute) information of data points without using any supervised (label) information during the training procedure. Representative unsupervised methods include iterative quantization (ITQ) [Gong and Lazebnik, 2011], isotropic hashing (IsoHash) [Kong and Li, 2012], discrete graph hashing (DGH) [Liu *et al.*, 2014], and scalable graph hashing (SGH) [Jiang and Li, 2015]. Supervised methods try to utilize supervised (label) information to learn the hash codes. The supervised information can be given in three different forms: *point-wise labels*, *pairwise labels* and *ranking labels*. Representative point-wise label based methods include CCA-ITQ [Gong and Lazebnik, 2011], supervised discrete hashing (SDH) [Shen *et al.*, 2015] and the deep hashing method in [Lin *et al.*, 2015]. Representative pairwise label based methods include sequential projection learning for hashing (SPLH) [Wang *et al.*, 2010], minimal loss hashing (MLH) [Norouzi and Fleet, 2011], supervised hashing with kernels (KSH) [Liu *et al.*, 2012], two-step hashing (TSH) [Lin *et al.*, 2013], fast supervised hashing (FastH) [Lin *et al.*, 2014], latent factor hashing (LFH) [Zhang *et al.*, 2014], convolutional neural network hashing (CNNH) [Xia *et al.*, 2014], and column sampling based discrete supervised hashing (COSDISH) [Kang *et al.*, 2016]. Representative ranking label based methods include ranking-based supervised hashing (RSH) [Wang *et al.*, 2013b], column generation hashing (CGHash) [Li *et al.*, 2013], order preserving hashing (OPH) [Wang *et al.*, 2013a], ranking preserving hashing (RPH) [Wang *et al.*, 2015], and some deep hashing methods [Zhao *et al.*, 2015a; Lai *et al.*, 2015; Zhang *et al.*, 2015]

Although a lot of hashing methods have been proposed as shown above, most existing hashing methods, including some deep hashing methods [Salakhutdinov and Hinton, 2009;

Masci *et al.*, 2014; Liong *et al.*, 2015], are based on hand-crafted features. In these methods, the hand-crafted feature construction procedure is independent of the hash-code and hash function learning procedure, and then the resulted features might not be optimally compatible with the hashing procedure. Hence, these existing hand-crafted feature based hashing methods might not achieve satisfactory performance in practice. To overcome the shortcoming of existing hand-crafted feature based methods, some feature learning based deep hashing methods [Zhao *et al.*, 2015a; Lai *et al.*, 2015; Zhang *et al.*, 2015] have recently been proposed to perform simultaneous feature learning and hash-code learning with deep neural networks, which have shown better performance than traditional hashing methods with hand-crafted features. Most of these deep hashing methods are supervised whose supervised information is given with triplet labels which are a special case of ranking labels.

For another common application scenario with pairwise labels, there have appeared few feature learning based deep hashing methods. To the best of our knowledge, CNNH [Xia *et al.*, 2014] is the only one which adopts deep neural network, which is actually a convolutional neural network (CNN) [LeCun *et al.*, 1989], to perform feature learning for supervised hashing with pairwise labels. CNNH is a two-stage method. In the first stage, the hash codes are learned from the pairwise labels, and then the second stage tries to learn the hash function and feature representation from image pixels based on the hash codes from the first stage. In CNNH, the learned feature representation in the second stage cannot give feedback for learning better hash codes in the first stage. Hence, CNNH cannot perform simultaneous feature learning and hash-code learning, which still has limitations. This has been verified by the authors of CNNH themselves in another paper [Lai *et al.*, 2015].

In this paper, we propose a novel deep hashing method, called *deep pairwise-supervised hashing* (DPSH), for applications with pairwise labels. The main contributions of DPSH are outlined as follows:

- DPSH is an end-to-end learning framework which contains three key components. The first component is a deep neural network to learn image representation from pixels. The second component is a hash function to map the learned image representation to hash codes. And the third component is a loss function to measure the quality of hash codes guided by the pairwise labels. All the three components are seamlessly integrated into the same deep architecture to map the images from pixels to the pairwise labels in an end-to-end way. Hence, different components can give feedback to each other in DPSH, which results in learning better codes than other methods without end-to-end architecture.
- To the best of our knowledge, DPSH is the first method which can perform simultaneous feature learning and hash-code learning for applications with pairwise labels.
- Experiments on real datasets show that DPSH can outperform other methods to achieve the state-of-the-art performance in image retrieval applications.

2 Notation and Problem Definition

2.1 Notation

We use boldface lowercase letters like $\mathbf{z}$ to denote vectors. Boldface uppercase letters like $\mathbf{Z}$ are used to denote matrices. The transpose of $\mathbf{Z}$ is denoted as $\mathbf{Z}^T$. $\|\cdot\|_2$ is used to denote the Euclidean norm of a vector. $sgn(\cdot)$ denotes the element-wise sign function which returns 1 if the element is positive and returns -1 otherwise.

2.2 Problem Definition

Suppose we have n points (images) $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ where $\mathbf{x}_i$ is the feature vector of point i. $\mathbf{x}_i$ can be the hand-crafted features or the raw pixels in image retrieval applications. The specific meaning of $\mathbf{x}_i$ can be easily determined from the context. Besides the feature vectors, the training set of supervised hashing with pairwise labels also contains a set of pairwise labels $\mathcal{S} = \{s_{ij}\}$ with $s_{ij} \in \{0, 1\}$, where $s_{ij} = 1$ means that $\mathbf{x}_i$ and $\mathbf{x}_j$ are similar, $s_{ij} = 0$ means that $\mathbf{x}_i$ and $\mathbf{x}_j$ are dissimilar. Here, the pairwise labels typically refer to semantic labels provided with manual effort.

The goal of supervised hashing with pairwise labels is to learn a binary code $\mathbf{b}_i \in \{-1, 1\}^c$ for each point $\mathbf{x}_i$, where c is the code length. The binary codes $\mathcal{B} = \{\mathbf{b}_i\}_{i=1}^n$ should preserve the similarity in $\mathcal{S}$. More specifically, if $s_{ij} = 1$, the binary codes $\mathbf{b}_i$ and $\mathbf{b}_j$ should have a low Hamming distance. Otherwise if $s_{ij} = 0$, the binary codes $\mathbf{b}_i$ and $\mathbf{b}_j$ should have a high Hamming distance. In general, we can write the binary code as $\mathbf{b}_i = h(\mathbf{x}_i) = [h_1(\mathbf{x}_i), h_2(\mathbf{x}_i), \cdots, h_c(\mathbf{x}_i)]^T$, where $h(\mathbf{x}_i)$ is the hash function to learn.

3 Model and Learning

Most existing pairwise label based supervised hashing methods, including SPLH [Wang *et al.*, 2010], MLH [Norouzi and Fleet, 2011], KSH [Liu *et al.*, 2012], TSH [Lin *et al.*, 2013], FastH [Lin *et al.*, 2014], and LFH [Zhang *et al.*, 2014], adopt hand-crafted features for hash function learning. As stated in Section 1, these methods cannot achieve satisfactory performance because the hand-crafted features might not be optimally compatible with the hash function learning procedure. CNNH [Xia *et al.*, 2014] adopts CNN to perform feature learning from raw pixels. However, CNNH is a two-stage method which cannot perform simultaneous feature learning and hash-code learning in an end-to-end way.

In this section, we introduce our model, called *deep pairwise-supervised hashing* (DPSH), which can perform simultaneous feature learning and hash-code learning in an end-to-end framework.

3.1 Model

Figure 1 shows the end-to-end deep learning architecture for our DPSH model, which contains the feature learning part and the objective function part.

Feature Learning Part

Our DPSH model contains a CNN model from [Chatfield *et al.*, 2014] as a component. More specifically, the feature learning part has seven layers which are the same as those of CNN-F in [Chatfield *et al.*, 2014]. Other CNN architectures,

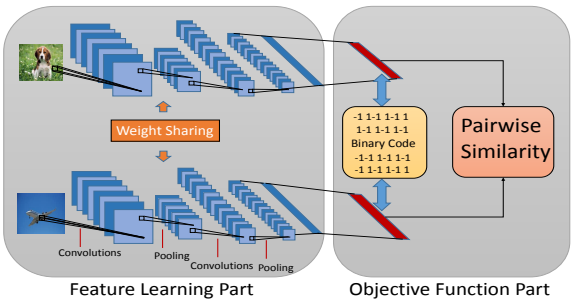


Figure 1: The end-to-end deep learning architecture for DPSH.

such as the AlexNet [Krizhevsky *et al.*, 2012], can also be used to substitute the CNN-F network in DPSH. But it is not the focus of this paper to study different networks. Hence, we just use CNN-F for illustrating the effectiveness of our DPSH model, and leave the study of other candidate networks for future pursuit. Please note that there are two CNNs (top CNN and bottom CNN) in Figure 1. These two CNNs have the same structure and share the same weights. That is to say, both the input and loss function are based on pairs of images.

The detailed configuration of the feature learning part of our DPSH model is shown in Table 1. More specifically, it contains 5 convolutional layers (conv 1-5) and 2 fully-connected layers (full 6-7). Each convolutional layer is described in several aspects: "filter" specifies the number of convolution filters and their receptive field size, denoted as "num x size x size"; "stride" indicates the convolution stride which is the interval at which to apply the filters to the input; "pad" indicates the number of pixels to add to each side of the input; "LRN" indicates whether Local Response Normalization (LRN) [Krizhevsky *et al.*, 2012] is applied; "pool" indicates the downsampling factor. "4096" in the fully-connected layer indicates the dimensionality of the output. The activation function for all layers is the REctification Linear Unit (RELU) [Krizhevsky *et al.*, 2012].

Table 1: Configuration of the feature learning part in DPSH.

Layer	Configuration
conv1	filter 64x11x11, stride 4x4, pad 0, LRN, pool 2x2
conv2	filter 256x5x5, stride 1x1, pad 2, LRN, pool 2x2
conv3	filter 256x3x3, stride 1x1, pad 1
conv4	filter 256x3x3, stride 1x1, pad 1
conv5	filter 256x3x3, stride 1x1, pad 1, pool 2x2
full6	4096
full7	4096

Objective Function Part

Given the binary codes $\mathcal{B} = \{\mathbf{b}_i\}_{i=1}^n$ for all the points, we can define the likelihood of the pairwise labels $\mathcal{S} = \{s_{ij}\}$ as that of LFH [Zhang *et al.*, 2014]:

$$p(s_{ij}|\mathcal{B}) = \begin{cases} \sigma(\Omega_{ij}), & s_{ij} = 1 \\ 1 - \sigma(\Omega_{ij}), & s_{ij} = 0 \end{cases}$$

where $\Omega_{ij} = \frac{1}{2}\mathbf{b}_i^T\mathbf{b}_j$, and $\sigma(\Omega_{ij}) = \frac{1}{1+e^{-\Omega_{ij}}}$. Please note that $\mathbf{b}_i \in \{-1, 1\}^c$.

By taking the negative log-likelihood of the observed pairwise labels in $\mathcal{S}$, we can get the following optimization problem:

$$\begin{aligned} \min_{\mathcal{B}} \mathcal{J}_1 &= -\log p(\mathcal{S}|\mathcal{B}) = -\sum_{s_{ij}\in\mathcal{S}} \log p(s_{ij}|\mathcal{B}) \\ &= -\sum_{s_{ij}\in\mathcal{S}} (s_{ij}\Omega_{ij} - \log(1 + e^{\Omega_{ij}})). \end{aligned} \quad (1)$$

It is easy to find that the above optimization problem can make the Hamming distance between two similar points as small as possible, and simultaneously make the Hamming distance between two dissimilar points as large as possible. This exactly matches the goal of supervised hashing with pairwise labels.

The problem in (1) is a discrete optimization problem, which is hard to solve. LFH [Zhang *et al.*, 2014] solves it by directly relaxing $\{\mathbf{b}_i\}$ from discrete to continuous, which might not achieve satisfactory performance [Kang *et al.*, 2016].

In this paper, we design a novel strategy which can solve the problem in (1) in a discrete way. First, we reformulate the problem in (1) as the following equivalent one:

$$\begin{aligned} \min_{\mathcal{B},\mathcal{U}} \mathcal{J}_2 &= -\sum_{s_{ij}\in\mathcal{S}} (s_{ij}\Theta_{ij} - \log(1 + e^{\Theta_{ij}})) \\ s.t. \quad & \mathbf{u}_i = \mathbf{b}_i, \quad \forall i = 1, 2, \cdots, n \\ & \mathbf{u}_i \in \mathbb{R}^{c\times 1}, \quad \forall i = 1, 2, \cdots, n \\ & \mathbf{b}_i \in \{-1, 1\}^c, \quad \forall i = 1, 2, \cdots, n \end{aligned} \quad (2)$$

where $\Theta_{ij} = \frac{1}{2}\mathbf{u}_i^T\mathbf{u}_j$, and $\mathcal{U} = \{\mathbf{u}_i\}_{i=1}^n$.

To optimize the problem in (2), we can optimize the following regularized problem by moving the equality constraints in (2) to the regularization terms:

$$\begin{aligned} \min_{\mathcal{B},\mathcal{U}} \mathcal{J}_3 = &-\sum_{s_{ij}\in\mathcal{S}} (s_{ij}\Theta_{ij} - \log(1 + e^{\Theta_{ij}})) \\ &+ \eta \sum_{i=1}^{n} \|\mathbf{b}_i - \mathbf{u}_i\|_2^2, \end{aligned}$$

where η is the regularization term (hyper-parameter).

DPSH Model

To integrate the above feature learning part and objective function part into an end-to-end framework, we set

$$\mathbf{u}_i = \mathbf{W}^T\phi(\mathbf{x}_i; \theta) + \mathbf{v},$$

where θ denotes all the parameters of the seven layers in the feature learning part, $\phi(\mathbf{x}_i; \theta)$ denotes the output of the full7 layer associated with point $\mathbf{x}_i$, $\mathbf{W} \in \mathbb{R}^{4096\times c}$ denotes a weight matrix, $\mathbf{v} \in \mathbb{R}^{c\times 1}$ is a bias vector. It means that we connect the feature learning part and the objective function part into the same framework by a fully-connected layer, with the weight matrix $\mathbf{W}$ and bias vector $\mathbf{v}$. After connecting the two parts, the problem for learning becomes:

$$\min_{\mathcal{B},\mathbf{W},\mathbf{v},\theta} \mathcal{J} = -\sum_{s_{ij}\in\mathcal{S}} (s_{ij}\Theta_{ij} - \log(1+e^{\Theta_{ij}})) + \eta\sum_{i=1}^{n} \|\mathbf{b}_i - (\mathbf{W}^T\phi(\mathbf{x}_i;\theta)+\mathbf{v})\|_2^2. \quad (3)$$

As a result, we get an end-to-end deep hashing model, called DPSH, to perform simultaneous feature learning and hash-code learning in the same framework.

3.2 Learning

In the DPSH model, the parameters for learning contain $\mathbf{W}$, $\mathbf{v}$, θ and $\mathcal{B}$. We adopt a minibatch-based strategy for learning. More specifically, in each iteration we sample a minibatch of points from the whole training set, and then perform learning based on these sampled points.

We design an alternating method for learning. That is to say, we optimize one parameter with other parameters fixed.

The $\mathbf{b}_i$ can be directly optimized as follows:

$$\mathbf{b}_i = sgn(\mathbf{u}_i) = sgn(\mathbf{W}^T\phi(\mathbf{x}_i;\theta)+\mathbf{v}). \quad (4)$$

For the other parameters $\mathbf{W}$, $\mathbf{v}$ and θ, we use back-propagation (BP) for learning. In particular, we can compute the derivatives of the loss function with respect to $\mathbf{u}_i$ as follows:

$$\frac{\partial\mathcal{J}}{\partial\mathbf{u}_i} = \frac{1}{2}\sum_{j:s_{ij}\in\mathcal{S}}(a_{ij}-s_{ij})\mathbf{u}_j + \frac{1}{2}\sum_{j:s_{ji}\in\mathcal{S}}(a_{ji}-s_{ji})\mathbf{u}_j + 2\eta(\mathbf{u}_i-\mathbf{b}_i),$$

where $a_{ij} = \sigma(\frac{1}{2}\mathbf{u}_i^T\mathbf{u}_j)$.

Then, we can update the parameters $\mathbf{W}$, $\mathbf{v}$ and θ by utilizing back propagation:

$$\frac{\partial\mathcal{J}}{\partial\mathbf{W}} = \phi(\mathbf{x}_i;\theta)(\frac{\partial\mathcal{J}}{\partial\mathbf{u}_i})^T, \quad (5)$$

$$\frac{\partial\mathcal{J}}{\partial\mathbf{v}} = \frac{\partial\mathcal{J}}{\partial\mathbf{u}_i}, \quad (6)$$

$$\frac{\partial\mathcal{J}}{\partial\phi(\mathbf{x}_i;\theta)} = \mathbf{W}\frac{\partial\mathcal{J}}{\partial\mathbf{u}_i}. \quad (7)$$

The whole learning algorithm of DPSH is briefly summarized in Algorithm 1.

3.3 Out-of-Sample Extension

After we have completed the learning procedure, we can only get the hash codes for points in the training data. We still need to perform out-of-sample extension to predict the hash codes for the points which are not appeared in the training set.

The deep hashing framework of DPSH can be naturally applied for out-of-sample extension. For any point $\mathbf{x}_q \notin \mathcal{X}$, we can predict its hash code just by forward propagation:

$$\mathbf{b}_q = h(\mathbf{x}_q) = sgn(\mathbf{W}^T\phi(\mathbf{x}_q;\theta)+\mathbf{v}). \quad (8)$$

4 Experiment

All our experiments for DPSH are completed with MatConvNet [Vedaldi and Lenc, 2015] on a NVIDIA K80 GPU server. Our model can be trained at the speed of about 290 images per second with a single K80 GPU.

Algorithm 1 Learning algorithm for DPSH.

Input:
Training images $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ and a set of pairwise labels $\mathcal{S} = \{s_{ij}\}$.
Output:
The parameters $\mathbf{W}$, $\mathbf{v}$, θ and $\mathcal{B}$.
Initialization: Initialize θ with the CNN-F model; Initialize each entry of $\mathbf{W}$ and $\mathbf{v}$ by randomly sampling from a Gaussian distribution with mean 0 and variance 0.01.
REPEAT
Randomly sample a minibatch of points from $\mathcal{X}$, and for each sampled point $\mathbf{x}_i$, perform the following operations:

- Calculate $\phi(\mathbf{x}_i;\theta)$ by forward propagation;
- Compute $\mathbf{u}_i = \mathbf{W}^T\phi(\mathbf{x}_i;\theta)+\mathbf{v}$;
- Compute the binary code of $\mathbf{x}_i$ with $\mathbf{b}_i = sgn(\mathbf{u}_i)$.
- Compute derivatives for point $\mathbf{x}_i$ according to (5), (6) and (7);
- Update the parameters $\mathbf{W}, \mathbf{v}, \theta$ by utilizing back propagation;

UNTIL a fixed number of iterations

4.1 Datasets and Setting

We compare our model with several baselines on two widely used benchmark datasets: *CIFAR-10* and *NUS-WIDE*.

The CIFAR-10 [Krizhevsky, 2009] dataset consists of 60,000 32×32 color images which are categorized into 10 classes (6000 images per class). It is a single-label dataset in which each image belongs to one of the ten classes.

The NUS-WIDE dataset [Chua *et al.*, 2009; Zhao *et al.*, 2015b] has nearly 270,000 images collected from the web. It is a multi-label dataset in which each image is annotated with one or mutiple class labels from 81 classes. Following [Lai *et al.*, 2015], we only use the images associated with the 21 most frequent classes. For these classes, the number of images of each class is at least 5000.

We compare our method with several state-of-the-art hashing methods. These methods can be categorized into five classes:

- Unsupervised hashing methods with hand-crafted features, including SH [Weiss *et al.*, 2008] and ITQ [Gong and Lazebnik, 2011].
- Supervised hashing methods with hand-crafted features, including SPLH [Wang *et al.*, 2010], KSH [Liu *et al.*, 2012], FastH [Lin *et al.*, 2014], LFH [Zhang *et al.*, 2014], and SDH [Shen *et al.*, 2015].
- The above unsupervised methods and supervised methods with deep features extracted by the CNN-F of the feature learning part in our DPSH.
- Deep hashing methods with pairwise labels, including CNNH [Xia *et al.*, 2014].
- Deep hashing methods with triplet labels, including network in network hashing (NINH) [Lai *et al.*, 2015], deep semantic ranking based hashing (DSRH) [Zhao *et al.*, 2015a], deep similarity comparison hashing (DSCH) [Zhang *et al.*, 2015] and deep regularized

similarity comparison hashing (DRSCH) [Zhang *et al.*, 2015].

For hashing methods which use hand-crafted features, we represent each image in CIFAR-10 by a 512-dimensional GIST vector. And we represent each image in NUS-WIDE by a 1134-dimensional low level feature vector, including 64-D color histogram, 144-D color correlogram, 73-D edge direction histogram, 128-D wavelet texture, 225-D block-wise color moments and 500-D SIFT features.

For deep hashing methods, we first resize all images to be 224×224 pixels and then directly use the raw image pixels as input. We adopt the CNN-F network which has been pre-trained on the ImageNet dataset [Russakovsky *et al.*, 2014] to initialize the first seven layers of our DPSH framework. Similar initialization strategy has also been adopted by other deep hashing methods [Zhao *et al.*, 2015a].

As most existing hashing methods, the mean average precision (MAP) is used to measure the accuracy of our proposed method and other baselines. The hyper-parameter η in DPSH is chosen by a validation set, which is 10 for CIFAR-10 and 100 for NUS-WIDE unless otherwise stated.

4.2 Accuracy

Following [Xia *et al.*, 2014; Lai *et al.*, 2015], we randomly select 1000 images (100 images per class) as the query set in CIFAR-10. For the unsupervised methods, we use the rest images as the training set. For the supervised methods, we randomly select 5000 images (500 images per class) from the rest images as the training set. The pairwise label set $\mathcal{S}$ is constructed based on the image class labels. That is to say, two images will be considered to be similar if they share the same class label.

In NUS-WIDE, we randomly sample 2100 query images from 21 most frequent labels (100 images per class) by following the strategy in [Xia *et al.*, 2014; Lai *et al.*, 2015]. For supervised methods, we randomly select 500 images per class from the rest images as the training set. The pairwise label set $\mathcal{S}$ is constructed based on the image class labels. That is to say, two images will be considered to be similar if they share at least one common label. For NUS-WIDE, we calculate the MAP values within the top 5000 returned neighbors.

The MAP results are reported in Table 2, where DPSH, DPSH0, NINH and CNNH are deep methods, and all the other methods are non-deep methods with hand-crafted features. The result of NINH, CNNH, KSH and ITQ are from [Xia *et al.*, 2014; Lai *et al.*, 2015]. Please note that the above experimental setting and evaluation metric is exactly the same as that in [Xia *et al.*, 2014; Lai *et al.*, 2015]. Hence, the comparison is reasonable. We can find that our method DPSH dramatically outperform other baselines[1], including unsupervised methods, supervised methods with hand-crafted features, and deep hashing methods with feature learning.

[1]The accuracy of LFH in Table 2 is much lower than that in [Zhang *et al.*, 2014; Kang *et al.*, 2016] because less points are adopted for training in this paper. Please note that LFH is an efficient method which can be used for training large-scale supervised hashing problems. But the training efficiency is not the focus of this paper.

Both DPSH and CNNH are deep hashing methods with pairwise labels. By comparing DPSH to CNNH, we can find that the model (DPSH) with simultaneous feature learning and hash-code learning can outperform the other model (CNNH) without simultaneous feature learning and hash-code learning.

NINH is a triplet label based method. Although NINH can perform simultaneous feature learning and hash-code learning, it is still outperformed by DPSH. More comparison with triplet label based methods will be provided in Section 4.4.

To further verify the importance of simultaneous feature learning and hash-code learning, we design a variant of DPSH, called DPSH0, which does not update the parameter of the first seven layers (CNN-F layers) during learning. Hence, DPSH0 just uses CNN-F for feature extraction, and then adopts the extracted features to learn hash functions. The hash function learning procedure gives no feedback to the feature extraction procedure. By comparing DPSH to DPSH0, we find that DPSH can dramatically outperform DPSH0. It means that integrating feature learning and hash-code learning into the same framework in an end-to-end way can get a better solution than that without end-to-end learning.

4.3 Comparison to Non-Deep Baselines with Deep Features

To further verify the effectiveness of simultaneous feature learning and hash-code learning, we compare DPSH to other non-deep methods with deep features extracted by the CNN-F pre-trained on ImageNet. The results are reported in Table 3, where "FastH+CNN" denotes the FastH method with deep features and other methods have similar notations. We can find that our DPSH outperforms all the other non-deep baselines with deep features.

4.4 Comparison to Baselines with Ranking Labels

Most existing deep supervised hashing methods are based on ranking labels, especially triplet labels. Although the learning procedure of these methods is based on ranking labels, the learned model can also be used for evaluation scenario with pairwise labels. In fact, most triplet label based methods adopt pairwise labels as ground truth for evaluation [Lai *et al.*, 2015; Zhang *et al.*, 2015]. In Section 4.2, we have shown that our DPSH can outperform NINH. In this subsection, we will perform further comparison to other deep hashing methods with ranking labels (triplet labels). These methods include DSRH [Zhao *et al.*, 2015a], DSCH [Zhang *et al.*, 2015] and DRSCH [Zhang *et al.*, 2015].

The experimental setting in DSCH and DRSCH [Zhang *et al.*, 2015] is different from that in Section 4.2. To perform fair comparison, we adopt the same setting as that in [Zhang *et al.*, 2015] for evaluation. More specifically, in CIFAR-10 dataset, we randomly sample 10,000 query images (1000 images per class) and use the rest as the training set. In the NUS-WIDE dataset, we randomly sample 2100 query images from 21 most frequently happened semantic labels (100 images per class), and use the rest as training samples. For NUS-WIDE, the MAP values within the top 50,000 returned neighbors are used for evaluation.

Table 2: Accuracy in terms of MAP. The best MAPs for each category are shown in boldface. Here, the MAP value is calculated based on the top 5000 returned neighbors for NUS-WIDE dataset.

Method	CIFAR-10 (MAP)				NUS-WIDE (MAP)			
	12-bits	24-bits	32-bits	48-bits	12-bits	24-bits	32-bits	48-bits
DPSH	**0.713**	**0.727**	**0.744**	**0.757**	**0.794**	**0.822**	**0.838**	**0.851**
DPSH0	0.479	0.472	0.470	0.495	0.747	0.751	0.763	0.776
NINH	0.552	0.566	0.558	0.581	0.674	0.697	0.713	0.715
CNNH	0.439	0476	0.472	0.489	0.611	0.618	0.625	0.608
FastH	0.305	0.349	0.369	0.384	0.621	0.650	0.665	0.687
SDH	0.285	0.329	0.341	0.356	0.568	0.600	0.608	0.637
KSH	0.303	0.337	0.346	0.356	0.556	0.572	0.581	0.588
LFH	0.176	0.231	0.211	0.253	0.571	0.568	0.568	0.585
SPLH	0.171	0.173	0.178	0.184	0.568	0.589	0.597	0.601
ITQ	0.162	0.169	0.172	0.175	0.452	0.468	0.472	0.477
SH	0.127	0.128	0.126	0.129	0.454	0.406	0.405	0.400

Table 3: Accuracy in terms of MAP. The best MAPs for each category are shown in boldface. Here, the MAP value is calculated based on the top 5000 returned neighbors for NUS-WIDE dataset.

Method	CIFAR-10 (MAP)				NUSWIDE (MAP)			
	12-bits	24-bits	32-bits	48-bits	12-bits	24-bits	32-bits	48-bits
DPSH	**0.713**	**0.727**	**0.744**	**0.757**	**0.794**	**0.822**	**0.838**	**0.851**
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	0.790	0.799
LFH + CNN	0.208	0.242	0.266	0.339	0.695	0.734	0.739	0.759
SPLH + CNN	0.299	0.330	0.335	0.330	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

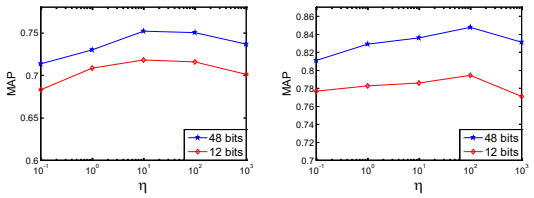
Table 4: Accuracy in terms of MAP. The best MAPs for each category are shown in boldface. Here, the MAP value is calculated based on the top 50,000 returned neighbors for NUS-WIDE dataset.

Method	CIFAR-10 (MAP)				NUS-WIDE (MAP)			
	16-bits	24-bits	32-bits	48-bits	16-bits	24-bits	32-bits	48-bits
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	0.620	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

The experimental results are shown in Table 4. Please note that the results of DPSH in Table 4 are different from those in Table 2, because the experimental settings are different. The results of DSRH, DSCH and DRSCH are directly from [Zhang *et al.*, 2015]. From Table 4, we can find that DPSH with pairwise labels can also dramatically outperform other baselines with triplet labels. Please note that DSRH, DSCH and DRSCH can also perform simultaneously feature learning and hash-code learning in an end-to-end framework.

4.5 Sensitivity to Hyper-Parameter

Figure 2 shows the effect of the hyper-parameter η. We can find that DPSH is not sensitive to η in a large range. For example, DPSH can achieve good performance on both datasets with $10 \leq \eta \leq 100$.



(a) CIFAR-10 (b) NUS-WIDE

Figure 2: Sensitivity to hyper-parameter.

5 Conclusion

In this paper, we have proposed a novel deep hashing methods, called DPSH, for settings with pairwise labels. To the best of our knowledge, DPSH is the first method which can perform simultaneous feature learning and hash-code learning for applications with pairwise labels. Because different components in DPSH can give feedback to each other, DPSH can learn better codes than other methods without end-to-end architecture. Experiments on real datasets show that DPSH can outperform other methods to achieve the state-of-the-art performance in image retrieval applications.

6 Acknowledgements

This work is supported by the NSFC (61472182), the Fundamental Research Funds for the Central Universities (20620140510), and the Tencent Fund (2014320001013613).

References

[Andoni and Indyk, 2006] Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006.

[Chatfield *et al.*, 2014] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014.

[Chua *et al.*, 2009] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. NUS-WIDE: A real-world web image database from national university of singapore. In *CIVR*, 2009.

[Gong and Lazebnik, 2011] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824, 2011.

[He *et al.*, 2013] Kaiming He, Fang Wen, and Jian Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, pages 2938–2945, 2013.

[Jiang and Li, 2015] Qing-Yuan Jiang and Wu-Jun Li. Scalable graph hashing with feature transformation. In *IJCAI*, pages 2248–2254, 2015.

[Kang *et al.*, 2016] Wang-Cheng Kang, Wu-Jun Li, and Zhi-Hua Zhou. Column sampling based discrete supervised hashing. In *AAAI*, 2016.

[Kong and Li, 2012] Weihao Kong and Wu-Jun Li. Isotropic hashing. In *NIPS*, pages 1655–1663, 2012.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012.

[Krizhevsky, 2009] Alex Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, University of Toronto, 2009.

[Kulis and Grauman, 2009] Brian Kulis and Kristen Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, pages 2130–2137, 2009.

[Lai *et al.*, 2015] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015.

[LeCun *et al.*, 1989] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, R. E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[Li *et al.*, 2013] Xi Li, Guosheng Lin, Chunhua Shen, Anton van den Hengel, and Anthony R. Dick. Learning hash functions using column generation. In *ICML*, pages 142–150, 2013.

[Lin *et al.*, 2013] Guosheng Lin, Chunhua Shen, David Suter, and Anton van den Hengel. A general two-step approach to learning-based hashing. In *ICCV*, pages 2552–2559, 2013.

[Lin *et al.*, 2014] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, and David Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, pages 1971–1978, 2014.

[Lin *et al.*, 2015] Kevin Lin, Huei-Fang Yang, Jen-Hao Hsiao, and Chu-Song Chen. Deep learning of binary hash codes for fast image retrieval. In *CVPR Workshops*, pages 27–35, 2015.

[Liong *et al.*, 2015] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *CVPR*, pages 2475–2483, 2015.

[Liu *et al.*, 2012] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

[Liu *et al.*, 2014] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. Discrete graph hashing. In *NIPS*, pages 3419–3427, 2014.

[Masci *et al.*, 2014] Jonathan Masci, Alex M. Bronstein, Michael M. Bronstein, Pablo Sprechmann, and Guillermo Sapiro. Sparse similarity-preserving hashing. In *ICLR*, 2014.

[Norouzi and Fleet, 2011] Mohammad Norouzi and David J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, pages 353–360, 2011.

[Rastegari *et al.*, 2013] Mohammad Rastegari, Jonghyun Choi, Shobeir Fakhraei, Daume Hal, and Larry S. Davis. Predictable dual-view hashing. In *ICML*, pages 1328–1336, 2013.

[Russakovsky *et al.*, 2014] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[Salakhutdinov and Hinton, 2009] Ruslan Salakhutdinov and Geoffrey E. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009.

[Shen *et al.*, 2015] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *CVPR*, 2015.

[Vedaldi and Lenc, 2015] Andrea Vedaldi and Karel Lenc. MatConvNet – convolutional neural networks for MATLAB. In *ACM MM*, 2015.

[Wang *et al.*, 2010] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Sequential projection learning for hashing with compact codes. In *ICML*, pages 1127–1134, 2010.

[Wang *et al.*, 2013a] Jianfeng Wang, Jingdong Wang, Nenghai Yu, and Shipeng Li. Order preserving hashing for approximate nearest neighbor search. In *ACM MM*, pages 133–142, 2013.

[Wang *et al.*, 2013b] Jun Wang, Wei Liu, Andy X. Sun, and Yu-Gang Jiang. Learning hash codes with listwise supervision. In *ICCV*, pages 3032–3039, 2013.

[Wang *et al.*, 2015] Qifan Wang, Zhiwei Zhang, and Luo Si. Ranking preserving hashing for fast similarity search. In *IJCAI*, pages 3911–3917, 2015.

[Weiss *et al.*, 2008] Yair Weiss, Antonio Torralba, and Robert Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

[Xia *et al.*, 2014] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, pages 2156–2162, 2014.

[Zhang *et al.*, 2014] Peichao Zhang, Wei Zhang, Wu-Jun Li, and Minyi Guo. Supervised hashing with latent factor models. In *SIGIR*, pages 173–182, 2014.

[Zhang *et al.*, 2015] Ruimao Zhang, Liang Lin, Rui Zhang, Wangmeng Zuo, and Lei Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Transactions on Image Processing*, 24(12):4766–4779, 2015.

[Zhao *et al.*, 2015a] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, pages 1556–1564, 2015.

[Zhao *et al.*, 2015b] Xueyi Zhao, Xi Li, and Zhongfei (Mark) Zhang. Multimedia retrieval via deep learning to rank. *IEEE Signal Processing Letters*, 22(9):1487–1491, 2015.