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
Collaborative Innovation Center of Novel Software Technology and Industrialization
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 [5] 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 [26, 9, 28, 3, 18, 22, 6, 38, 4, 13, 25], especially in image retrieval applications.

Existing hashing methods can be divided into data-independent methods and data-dependent methods [9, 18, 3]. 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) [1] 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 [3, 18]. 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 [18, 3, 13, 25].

The L2H methods can be further divided into two categories [18, 3, 13, 25]: 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) [3], anchor graph hashing (AGH) [19], isotropic hashing (IsoHash) [7], and discrete graph hashing (DGH) [17]. 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 [3], supervised discrete hashing (SDH) [25] and the deep hashing method in [15]. Representative pairwise label based methods include sequential projection learning for hashing (SPLH) [29], minimal loss hashing (MLH) [21], supervised hashing with kernels (KSH) [18], two-step hashing (TSH) [14], fast supervised hashing (FastH) [13], latent factor hashing (LFH) [35], and convolutional neural network hashing (CNNH) [34]. Representative ranking label based methods include ranking-based supervised hashing (RSH) [30], column generation hashing (CGHash) [12], order preserving hashing (OPH) [31], ranking preserving hashing (RPH) [32], and some deep hashing methods [37, 10, 36]

Although a lot of hashing methods have been proposed as shown above, most existing hashing methods, including some deep hashing methods [24, 26, 20, 16], 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 [37, 10, 36] 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 [34] is the only one which adopts deep neural network, which is actually a convolutional neural network (CNN) [11], 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 [10].

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 our DPSH method can outperform other methods to achieve the state-of-the-art performance in image retrieval applications.

The rest of this paper is organized as follows. Section 2 introduces the problem definition of this paper. Section 3 presents the details of our DPSH model and the learning algorithm. Section 4 shows the experimental results, and Section 5 concludes this paper and points out some possible directions for future pursuit.

2. Notation and Problem Definition

2.1. Notation

We use boldface lowercase letters like $\mathbf{z}$ to denote vectors, and the ith element of $\mathbf{z}$ is denoted as z_i. Boldface uppercase letters like $\mathbf{Z}$ are used to denote matrices. The inverse of $\mathbf{Z}$ is denoted as $\mathbf{Z}^{-1}$, and the transpose of $\mathbf{Z}$ is denoted as $\mathbf{Z}^T$. $\|\cdot\|_F$ is used to denote the Frobenius norm of a vector or matrix. $sgn(\cdot)$ 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 [29], MLH [21], KSH [18], TSH [14], FastH [13], and LFH [35], 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 [34] 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. Please note that there are two CNNs (top CNN and bottom CNN) in Figure 1. However, these two CNNs are exactly the same, which means that both the structures and the parameters of the two CNNs are the same. We use two CNNs to get a better illustration for the cases with pairwise labels. That is to say, both the input and loss function are based on pairs of images. Our DPSH model contains a CNN model from [2] as a component. More specifically, our DPSH model has eight layers and the first seven layers are the same as those in VGG-F[1], which is called CNN-F in [2]. The eighth layer of DPSH is a fully-connected layer which is used to generate hash codes. Please note that other CNN architectures, such as the AlexNet [8], can also be used to substitute the VGG-F network in our DPSH model. But it is not the focus of this paper to study different networks. Hence, we just use VGG-F for illustrating the effectiveness of our DPSH model, and leave the study of other candidate networks for future pursuit.

The detailed configuration of our DPSH model is shown in Table 1. More specifically, DPSH contains 5 convolutional layers (conv 1-5) and 3 fully-connected layers (full 6-8). 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) [8] is applied; "pool" indicates the downsampling factor; "4096" in the fully-connected layer indicates the dimensionality of the output. The activation function for all weight layers (except for full8) is the Rectified Linear Unit (ReLU) [8].

Let θ denote all the parameters of the first seven layers of the DPSH architecture, and $\phi(\mathbf{x}_i;\theta)$ denote the output of the full7 layer associated with point $\mathbf{x}_i$. The full8 layer is a fully-connected layer with c nodes, and its output are binary codes which are defined as follows:

[1]VGG is an abbreviation for Visual Geometry Group at University of Oxford.

Table 1. Configuration of our DPSH architecture.

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
full8	hash-code length c

$$\mathbf{b}_i = h(\mathbf{x}_i) = sgn(\mathbf{W}^T\phi(\mathbf{x}_i;\theta)+\mathbf{v}) = sgn(\mathbf{u}_i), \quad (1)$$

where $\mathbf{W}\in\mathbb{R}^{4096\times c}$ denotes the weight matrix connecting the full7 layer and full8 layer, $\mathbf{v}\in\mathbb{R}^c$ is the bias vector, and we let $\mathbf{u}_i = \mathbf{W}^T\phi(\mathbf{x}_i;\theta)+\mathbf{v}$.

Based on the learned binary codes which are outputed by the full8 layer, we can define the pairwise loss function similar to that of LFH [35]:

$$\begin{aligned} L &= -\log p(\mathcal{S}|\mathcal{B}) = -\sum_{s_{ij}\in\mathcal{S}} \log p(s_{ij}|\mathcal{B}) \\ &= -\sum_{s_{ij}\in\mathcal{S}} (s_{ij}\Theta_{ij} - \log(1+e^{\Theta_{ij}})), \end{aligned} \quad (2)$$

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

As stated in [35], the pairwise loss function in (2) reflects the negative log-likelihood of the observed pairwise labels in $\mathcal{S}$. Minimizing this loss function will make the Hamming distance between two similar points as small as possible, and simultaneously make the Hamming distance between two dissimilar points as high as possible. This exactly matches the goal of supervised hashing with pairwise labels. Hence, good performance can be expected with this pairwise loss function.

3.2. Learning

To learn the DPSH model, we need to learn the parameters of θ and $\{\mathbf{W},\mathbf{v}\}$. Because the $sgn(\cdot)$ function is not continuous, we relax it for the binary code learning procedure. More specifically, we relax the objective function in (2) by taking $\Theta_{ij} = \frac{1}{2}\mathbf{u}_i^T\mathbf{u}_j$.

Stochastic gradient descent (SGD) is used to learn the parameters. In particular, in each iteration we sample a minibatch of points from the whole training set, and then perform learning based on these sampled points. We use back-propagation (BP) to learn the parameters. In particular, we can compute the derivatives of the loss function with

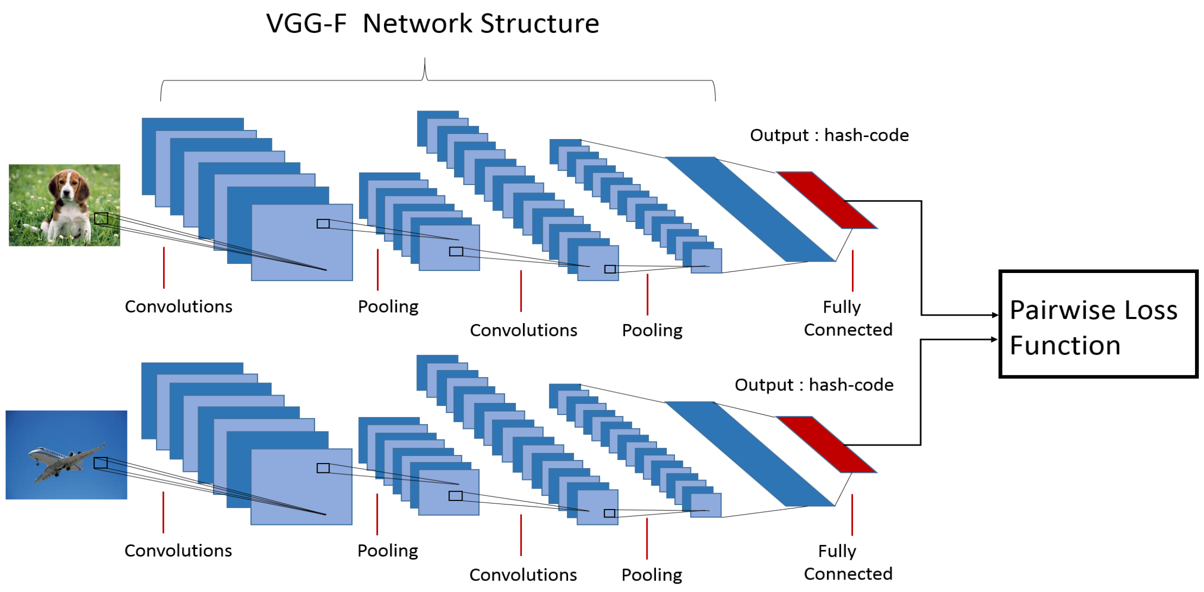


Figure 1. The end-to-end deep learning architecture for DPSH. Please note that there are two CNNs (top CNN and bottom CNN) in the figure. However, these two CNNs are exactly the same, which means than both the structures and the parameters of the two CNNs are the same. We use two CNNs to get a better illustration that the input and loss function are based on pairs of images.

respect to the (relaxed) hash codes as follows:

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{u}_i} =& \frac{1}{2}\sum_{j:s_{ij}\in\mathcal{S}}(a_{ij}-s_{ij})\mathbf{u}_j \\ &+\frac{1}{2}\sum_{j:s_{ji}\in\mathcal{S}}(a_{ji}-s_{ji})\mathbf{u}_j,\end{aligned} \quad (3)$$

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

These derivatives can be fed into the former layers of the network to update the parameters $\{\mathbf{W}, \mathbf{v}\}$ and then update θ though the BP learning rules. The whole learning algorithm of DPSH is briefly summarized in Algorithm 1.

3.3. Out-of-Sample Extension

After we have learned the hash codes and the hash function for the training data, we need to perform out-of-sample extension to predict the hash code 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 $\mathbf{x}_q$, we can predict its hash code just by forward propagation, i.e., $\mathbf{b}_q = sgn(\mathbf{W}^T\phi(\mathbf{x}_q;\theta)+\mathbf{v})$.

4. Experiment

All our experiments for DPSH are completed with MatConvNet [27] on a K40 GPU. MatConvNet can be run on NVIDIA GPUs as it includes CUDA implementations of most deep architectures. Our model can be trained at the speed of about 290 images per second with a single K40 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 network parameters $\theta$ and $\{\mathbf{W}, \mathbf{v}\}$.
Initialization: Initialize with the VGG-F model.
REPEAT
    Randomly sample a minibatch of images from $\mathcal{X}$, and for each sampled image $\mathbf{x}_i$, perform the following operations:
      • Calculate output $\mathbf{u}_i$ of image $\mathbf{x}_i$ by forward propagation.
      • Compute $\frac{\partial L}{\partial \mathbf{u}_i}$ for image $\mathbf{x}_i$ according to (3).
      • Update the parameters $\{\mathbf{W}, \mathbf{v}\}$ and then update $\theta$ by utilizing back propagation.
UNTIL a fix 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 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 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 [10, 18], 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 four classes:

- Unsupervised hashing methods with hand-crafted features, including SH [33] and ITQ [3].
- Supervised hashing methods with hand-crafted features, including SPLH [29], KSH [18], FastH [13], LFH [35], and SDH [25].
- Deep hashing methods with pairwise labels, including CNNH [34].
- Deep hashing methods with triplet labels, including network in network hashing (NINH) [10], deep semantic ranking based hashing (DSRH) [37], deep similarity comparison hashing (DSCH) [36] and deep regularized similarity comparison hashing (DRSCH) [36].

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 bag of words based on SIFT descriptions.

For deep hashing methods, we directly use the raw image pixels as input. We adopt the VGG-F network which has been pre-trained on the ImageNet dataset [23] to initialize the first seven layers of our DPSH network. Similar initialization strategy has also been adopted by other deep hashing methods [37].

As most existing hashing methods, the mean average precision (MAP) is used to measure the accuracy of our proposed method and other baselines.

4.2. Accuracy

Following [34, 10], 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 [34, 10]. 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 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. The result of NINH, CNNH, KSH and ITQ are from [10, 34]. Please note that the above experimental setting and evaluation metric is exactly the same as that in [34, 10]. Hence, the comparison is reasonable. We can find that our method DPSH dramatically outperform other baselines[2], including unsupervised methods, supervised methods with hand-crafted features, and deep hashing methods with feature learning.

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 learning of features and hash codes.

NINH is a triplet label based method. Although NINH can perform simultaneous feature learning and hash-code learning, it is still outperformed by our DPSH model. More comparison with triplet label based methods will be provided in Section 4.3.

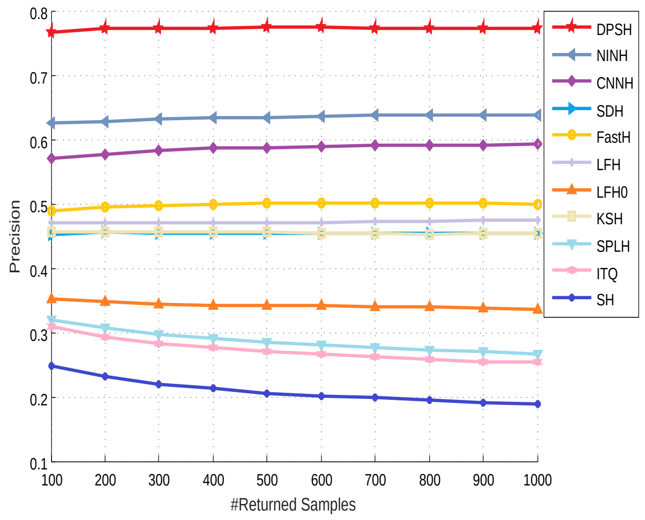
To further verify the importance of simultaneous feature learning and hash-code learning, we design a variant of our DPSH, denoted as DPSH0, which does not update the parameter of the first seven layers (VGG-F layers) during learning. Hence, DPSH0 just uses the VGG-F for feature extraction, and then based on the extracted features to learn hash functions. The hash function learning procedure will give no feedback to the feature extraction procedure. By comparing DPSH to DPSH0, we can 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.

Figure 2 shows the precision results on top returned samples with code-length being 48. Once again, we can find that our DPSH can dramatically outperform other baselines.

[2]Please note that LFH [35] is efficient to solve large-scale supervised hashing problems by using a (column) sampling strategy which is different from that of other methods. Hence, we adopt two variants of LFH in this paper. LFH0 refers to the variant with 5000 images for training, and LFH refers to the variant with all available images for training.

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	**0.682**	**0.686**	**0.725**	**0.733**	**0.785**	**0.810**	**0.824**	**0.833**
DPSH0	0.458	0.512	0.521	0.525	0.723	0.743	0.748	0.758
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	0.278	0.435	0.518	0.561	0.747	0.784	0.808	0.802
LFH0	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



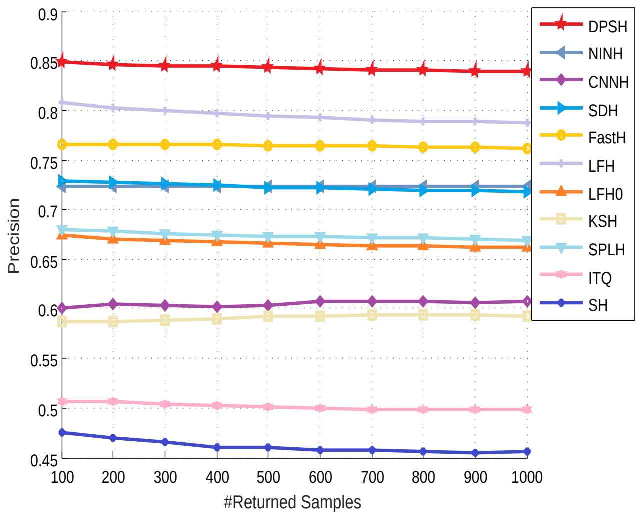


Figure 2. The precision results on top returned samples with code-length being 48. Left: CIFAR-10; Right: NUS-WIDE.

4.3. 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 [10, 36]. 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 [37], DSCH [36] and DRSCH [36].

The experimental setting in DSCH and DRSCH [36] is different from that in Section 4.2. To perform fair comparison, we adopt the same setting as that in [36] 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.

The experimental results are shown in Table 3. Please note that the results of our DPSH in Table 3 are different from those in Table 2, because the experimental settings are different. The results of DSRH, DSCH and DRSCH are directly from [36]. From Table 3, we can find that our 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.

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 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	**0.742**	**0.764**	**0.765**	**0.770**	**0.698**	**0.711**	**0.726**	**0.730**
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

4.4. Illustration

In Figure 3, we illustrate the retrieval results (top 15 returned images) for ten query images from CIFAR-10 using Hamming ranking on 48-bit hash code. The images with red rectangles indicate mistakes (images dissimilar to the query image). We can find that the returned results are very promising. This verifies that the deep learning framework can effectively capture the essential image features.

5. Conclusion

In this paper, we have proposed a novel deep hashing methods, called DPSH, for settings with pairwise labels. DPSH is an end-to-end learning framework which can simultaneously perform feature learning and hash-code learning. Because different components in DPSH can give feedback to each other, DPSH can learn 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 our DPSH method can outperform other methods, including both hand-crafted feature based methods and feature learning base methods, to achieve the state-of-the-art performance in image retrieval applications.

Future work will try to integrate better CNN based feature learning methods into our framework to get more promising results.

References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the Annual Symposium on Foundations of Computer Science*, pages 459–468, 2006. 1

[2] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014. 3

[3] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 817–824, 2011. 1, 5

[4] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2938–2945, 2013. 1

[5] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 604–613, 1998. 1

[6] Z. Jin, Y. Hu, Y. Lin, D. Zhang, S. Lin, D. Cai, and X. Li. Complementary projection hashing. In *IEEE International Conference on Computer Vision*, pages 257–264, 2013. 1

[7] W. Kong and W.-J. Li. Isotropic hashing. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 1655–1663, 2012. 1

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceeding of the Advances in Neural Information Processing Systems*, pages 1106–1114, 2012. 3

[9] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2130–2137, 2009. 1

[10] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3278, 2015. 1, 2, 5, 6

[11] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989. 2

[12] X. Li, G. Lin, C. Shen, A. van den Hengel, and A. R. Dick. Learning hash functions using column generation. In *Proceedings of the International Conference on Machine Learning*, pages 142–150, 2013. 1

[13] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1971–1978, 2014. 1, 2, 5

[14] G. Lin, C. Shen, D. Suter, and A. v. d. Hengel. A general two-step approach to learning-based hashing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2552–2559, 2013. 1, 2

[15] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 27–35, 2015. 1

[16] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *IEEE Con-*

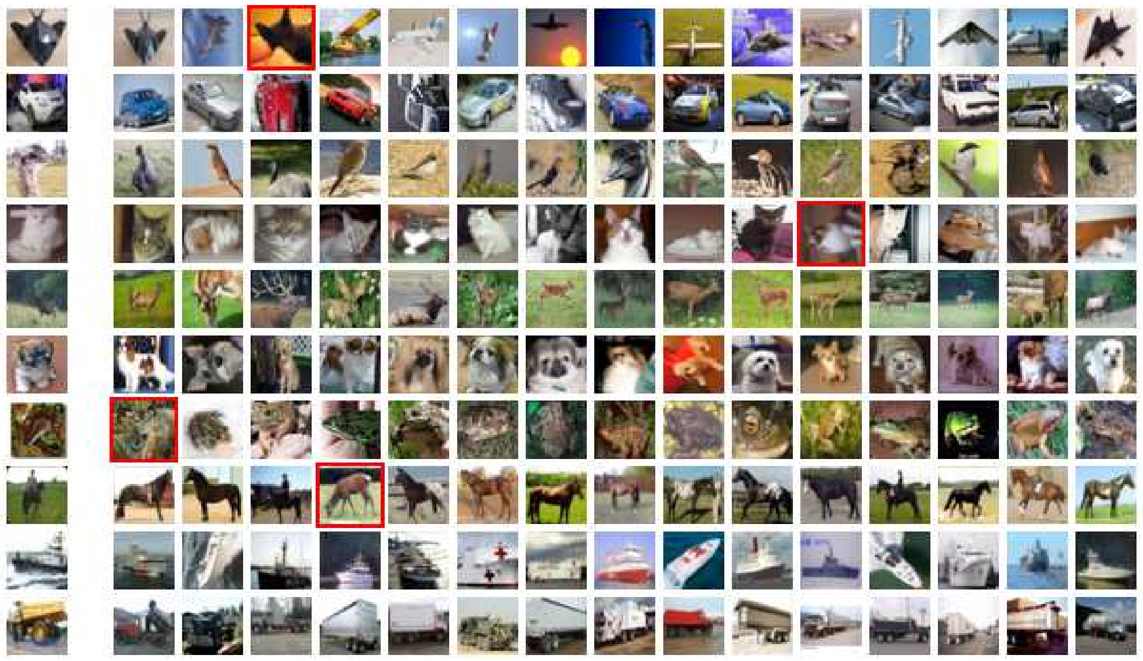


Figure 3. Retrieval results (top 15 returned images) for ten query images from CIFAR-10 using Hamming ranking on 48-bits hash code. Red rectangles indicate mistakes.

ference on Computer Vision and Pattern Recognition, pages 2475–2483, 2015. 1

[17] W. Liu, C. Mu, S. Kumar, and S. Chang. Discrete graph hashing. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 3419–3427, 2014. 1

[18] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2074–2081, 2012. 1, 2, 5

[19] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *Proceedings of the International Conference on Machine Learning*, 2011. 1

[20] J. Masci, A. M. Bronstein, M. M. Bronstein, P. Sprechmann, and G. Sapiro. Sparse similarity-preserving hashing. In *International Conference on Learning Representations*, 2014. 1

[21] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *Proceedings of the International Conference on Machine Learning*, pages 353–360, 2011. 1, 2

[22] M. Rastegari, J. Choi, S. Fakhraei, D. Hal, and L. S. Davis. Predictable dual-view hashing. In *Proceedings of the International Conference on Machine Learning*, pages 1328–1336, 2013. 1

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. 5

[24] R. Salakhutdinov and G. E. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009. 1

[25] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015. 1, 5

[26] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2008. 1

[27] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceeding of the ACM Int. Conf. on Multimedia*, 2015. 4

[28] J. Wang, O. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3424–3431, 2010. 1

[29] J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for hashing with compact codes. In *Proceedings of the International Conference on Machine Learning*, pages 1127–1134, 2010. 1, 2, 5

[30] J. Wang, W. Liu, A. X. Sun, and Y. Jiang. Learning hash codes with listwise supervision. In *IEEE International Conference on Computer Vision*, pages 3032–3039, 2013. 1

[31] J. Wang, J. Wang, N. Yu, and S. Li. Order preserving hashing for approximate nearest neighbor search. In *ACM Multimedia Conference*, pages 133–142, 2013. 1

[32] Q. Wang, Z. Zhang, and L. Si. Ranking preserving hashing for fast similarity search. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, pages 3911–3917, 2015. 1

[33] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proceedings of the Annual Conference on Neural Information Processing Systems*, pages 1753–1760, 2008. 5

[34] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2156–2162, 2014. 1, 2, 5

[35] P. Zhang, W. Zhang, W.-J. Li, and M. Guo. Supervised hashing with latent factor models. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 173–182, 2014. 1, 2, 3, 5

[36] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. 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. 1, 2, 5, 6

[37] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1556–1564, 2015. 1, 2, 5, 6

[38] X. Zhu, Z. Huang, H. Cheng, J. Cui, and H. T. Shen. Sparse hashing for fast multimedia search. *ACM Transactions on Information Systems*, 31(2):9, 2013. 1