

# Deep Asymmetric Pairwise Hashing

Fumin Shen<sup>1</sup>, Xin Gao<sup>1</sup>, Li Liu<sup>2</sup>, Yang Yang<sup>1</sup>, Heng Tao Shen<sup>1\*</sup>

<sup>1</sup>Center for Future Media and School of Computer Science and Engineering, University of Electronic Science and Technology of China

<sup>2</sup>School of Computing Sciences, University of East Anglia

## ABSTRACT

Recently, deep neural networks based hashing methods have greatly improved the multimedia retrieval performance by simultaneously learning feature representations and binary hash functions. Inspired by the latest advance in the asymmetric hashing scheme, in this work, we propose a novel Deep Asymmetric Pairwise Hashing approach (DAPH) for supervised hashing. The core idea is that two deep convolutional models are jointly trained such that their output codes for a pair of images can well reveal the similarity indicated by their semantic labels. A pairwise loss is elaborately designed to preserve the pairwise similarities between images as well as incorporating the independence and balance hash code learning criteria. By taking advantage of the flexibility of asymmetric hash functions, we devise an efficient alternating algorithm to optimize the asymmetric deep hash functions and high-quality binary code jointly. Experiments on three image benchmarks show that DAPH achieves the state-of-the-art performance on large-scale image retrieval.

## KEYWORDS

Binary code; asymmetric hashing; deep hashing

## 1 INTRODUCTION

The growing explosion of big data highlights the importance in designing efficient indexing and retrieval methods recently. Searching for similar data samples in a given database essentially relates to the fundamental problem of nearest neighbor search. Hashing technique has become a popular nearest neighbor search technique for image retrieval on large datasets [31]. Hashing methods are proposed to map images to compact binary codes that approximately preserve the data structure or semantic affinity in original space. These compact codes can substantially reduce the storage overhead and also speed up image search significantly.

Many hashing methods have been intensively studied and proposed e.g., [8, 13, 19, 27, 28, 32, 39, 44, 45] for several decades. Recently hashing research is focused on learning compact codes from available training data, a.k.a. *learning to hash*. Different from LSH, the goal of learning to hash is to learn data-dependent hash functions which generate more compact codes to achieve good search

accuracy. Hence L2H methods have become more and more popular than data-independent methods in real applications. L2H methods can be divided into unsupervised methods and supervised methods. Unsupervised hashing methods attempt to integrate the data properties, such as data distributions and manifold structures to design compact hash codes with improved accuracy. Representative unsupervised methods include spectral hashing (SH) [32], iterative quantization hashing (ITQ) [8], isotropic hashing (IsoH) [10], anchor graph hashing (AGH) [20], inductive manifold hashing (IMH) [28], etc. In addition, supervised methods can incorporate semantic labels or relevance to mitigate the semantic gap and has demonstrated superior image search accuracy compared with unsupervised hashing algorithms. Exemplar supervised hashing schemes include binary reconstructive embedding (BRE)[13], kernel-based supervised hashing (KSH) [19], supervised discrete hashing (SDH) [27]. Many other methods fall in this category [15, 22, 29, 35, 37].

The appealing property of supervised hashing methods is to minimize the semantic gap between the similarities given in the original space and in the hash coding space. In the hashing literature, pairwise similarity is widely adopted to make the distances of similarity pairs in original space and Hamming space as consistent as possible, i.e., semantically similar images should have similar binary codes. Many supervised learning paradigms have been explored using such pairwise semantic relationships to learn semantically relevant hash function and have proven that preserving pairwise similarity of data is important for achieving promising retrieval performance [13, 19, 22].

In the pipeline of most existing pairwise-preserving supervised hashing methods for images, each input image is firstly represented by a vector of traditional hand-crafted visual descriptors, followed by separate projection and quantization steps to encode image vector into binary codes. One shortcoming of these hand-crafted feature methods is that the feature extraction procedure, which means that the hand-crafted features might not achieve satisfactory performance with the binary code learning procedure. To overcome the shortcoming of existing hand-crafted feature based methods, some feature learning based deep hashing methods have been recently proposed to perform simultaneous feature learning and hash code learning with deep neural networks [9, 14, 16–18, 33, 36, 43]. However, these deep learning to hash methods is symmetric hashing scheme which similarity between two objects is approximated by the Hamming distance between the outputs of the same hash function. The crucial disadvantage of these symmetric hashing scheme is that the symmetric discrete constraint might be difficult to optimize and not necessary to learn binary codes of training data. Furthermore, recent literature [21] proposes using two distinct mappings to approximate the similarity affinity by the Hamming distance between two different hashing functions. They refer to such hashing schemes as “asymmetric hashing”. [21] has exhibited

\*Corresponding author: Heng Tao Shen

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).  
MM'17, October 23–27, 2017, Mountain View, CA, USA.  
© 2017 ACM. ISBN 978-1-4503-4906-2/17/10...\$15.00  
DOI: <https://doi.org/10.1145/3123266.3123345>

that the asymmetric discrete matrices can preserve more similarity (label) information, which usually means better retrieval accuracy.

Motivated by the aforementioned observations, in this work, we strive for the goal of designing efficient framework based on asymmetric hashing function and deep model for learning high-quality hash codes. To achieve this goal, we propose a novel Deep Asymmetric Pairwise hashing (DAPH). An overall view of the proposed framework is illustrated in Figure 1. Here we use two deep neural network to construct asymmetric hash functions to learn directly from images, which provides much information than hand-crafted features. A loss function defined on a set of pairs is designed for pairwise similarity-preserving learning. To further improve the quality of the hash functions, we also incorporate the independence and balance properties with the binary codes. The resulting problem with these binary constraints is NP-hard, which we solve by an efficient alternating algorithm optimizing over binary codes and hash functions in an iterative way. Our main contributions include:

- To our best knowledge, DAPH is the first deep asymmetric hashing method which integrates the feature learning and asymmetric hash functions learning into the end-to-end deep learning framework.
- DAPH aims to generate pairwise similarity-preserving binary codes as well as the semantic information rich asymmetric hash functions. By taking advantage of the asymmetric hash form, we are able to devise an efficient alternating optimization algorithm for the employed optimization problem.
- Experiments on several large datasets show the effectiveness of DAPH and its advantage over many recently proposed supervised hashing methods on image retrieval.

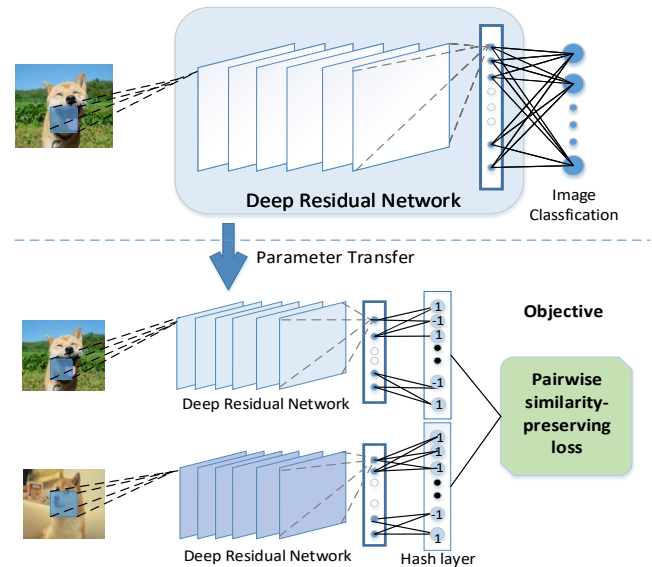
The rest of this paper is organized as follows. Section 2 discusses the related works to our method. The proposed approach is elaborated in details in Section 3. In section 4, we present the experimental evaluation on large-scale image datasets, followed by the conclusion in Section 5.

## 2 RELATED WORKS

As described before, existing learning to hash methods can be categorized into two categories: unsupervised hashing and supervised hashing.

Unsupervised methods only utilize the feature information of the training data to learn hash functions that can encode input data pairs to binary codes. For example, Spectral Hashing [24] produces hash codes through solving a continuously relaxed mathematical program similar to Laplacian Eigenmaps; Iterative Quantization [8] proposes to minimize the quantization error on projected image descriptors in order to alleviate the information loss caused by the discrepancy between the input feature space and Hamming space; Inductive Manifold Hashing [28] tries to generate nonlinear hash functions.

Supervised methods try to leverage supervised information to learn compact hash codes. Iterative quantization with canonical correlated analysis (CCA-ITQ) [8], the extension of ITQ, utilizes CCA with labels to reduce the dimensionality of input data and binaries the outcome through minimizing the quantization error. Minimal Loss Hashing [22] introduces a pairwise hinge-like loss function



**Figure 1: Overview of the proposed DAPH method. The training procedure includes two stages: the pre-training with the deep residual network (top) and the deep asymmetric hash model training (bottom).**

and minimizes its upper bound to learn similarity-preserving binary codes. Binary Reconstruction Embedding [13] pursues hash functions data points and the distances of corresponding hash codes. Supervised Hashing with Kernels [19] utilizes the algebraic equivalence between Hamming distance and code inner product and employs a kernel formulation for target hash functions.

Recent progress in image classification [12], object detection [7], face recognition [30] and many other vision tasks [4] demonstrates the impressive learning power of deep neural network. In these different tasks, the deep neural network can be considered as a feature extractor guided by the objective functions specifically designed for the individual tasks. The successful applications of deep learning in various tasks imply that the features learned by deep neural networks can well capture the underlying semantic structure of images in spite of significant appearance variations.

The earliest work in deep-learning-based hashing is Semantic Hashing [24], which builds a stacked Restricted Boltzmann Machines to discover hidden binary units. [33] proposed a two-step hashing strategy named CNNH. It first factorizes the data similarity matrix to obtain target binary code and then jointly use the target codes and image labels to guide the network parameter optimization. [14] improved the two-stage CNNH by proposing DNNH, a simultaneous feature learning and hash coding deep network. [5] presented a binary encoding network built with purely fully-connected layers.

Deep neural networks learn the image representation and hash codes in one stage so that representation learning and hash learning are tightly coupled to benefit each other. [16] proposed a novel

deep hashing method (DHN). DHN is the first end-to-end framework which can perform simultaneous feature learning and hash code learning for applications with pairwise labels. Nevertheless, DHN focused on learning symmetric hashing function. Compared to above deep hashing methods, we have two advantages: (1) We propose to use two deep neural networks to generate asymmetric hashing functions and learn efficient image representation simultaneously. (2) We adopt the independence and balance properties to the binary codes so as to generate high-quality binary code.

### 3 DEEP ASYMMETRIC PAIRWISE HASHING

In this section, we first describe the detailed formulation of the proposed objective function for learning deep asymmetric hash codes. Then the optimization of our target will be elaborated.

#### 3.1 Problem Definition

To help better understand this section, we first introduce some notation. We are given a training set of  $N$  images:  $X = \{x_1, x_2, \dots, x_n\} \in \mathbb{R}^{N \times d_1 \times d_2 \times 3}$ , where  $x_i$  is an image and  $d_1/d_2$  is the height/width of the corresponding image. Additionally, pairs of images are associated with similarity label  $s_{ij}$ . Here, we use  $S = \{s_{ij}\}$  to indicate the similarity of two images where  $s_{ij} = 1$  implies  $x_i$  and  $x_j$  are similar and  $s_{ij} = 0$  indicates  $x_i$  and  $x_j$  are dissimilar. The supervised similarity  $s_{ij}$  is typically defined by some semantic information such as class labels. Let  $b_i \in \{-1, 1\}^k$  be the  $k$ -bit hash codes of image  $x_i$ ;  $h_j \in \{-1, 1\}^k$  be the  $k$ -bit hash codes of images  $x_j$ . Our goal is to learn two nonlinear mapping from  $X$  to  $k$ -bit binary codes:  $\mathcal{F}_1 : x_i \rightarrow b_i$ ,  $\mathcal{F}_2 : x_j \rightarrow h_j$ . The two nonlinear hash mappings should preserve the pairwise similarity in the input space. More specifically, the binary codes  $b_i$  and  $h_j$  should have small Hamming distance if  $s_{ij} = 1$  otherwise if  $s_{ij} = 0$ .

#### 3.2 Deep Hash Functions

As previously stated, deep hash functions using deep neural networks has more powerful learning capability than hand-crafted features extracted in advance and thus is able to learn feature representations for semantic similarity search. In this work, we propose an architecture of deep convolution network designed for asymmetric hash functions. The whole model is shown in Figure 1. The network is comprised of two components: (1) An end-to-end framework including two deep residual networks to integrate feature learning part and binary code learning part. (2) A designed pairwise loss to preserve the pairwise similarity and generate the high-quality binary code with the desirable properties of independence and balancing.

This architecture accepts input images with the pairwise form and the input images are  $d_1 \times d_2 \times 3$  size. For binary code learning, we replace the top layer of the softmax classifier in the original ResNet50 with a new *fch* hash layer of  $k$  hidden units, which transform the final convolution representation to  $k$ -dimension binary code. Let us denote the response vector on the topmost layer of the first ResNet as  $u_i = \phi(x_i; \theta_1)$ , where  $\theta_1$  denotes the parameters of the first ResNet50 for learning feature,  $\phi(x_i; \theta_1)$  implicitly defines the highly non-linear mapping from the raw image  $x_i$  to binary-like code  $u_i$ ; the response vector on the topmost layer of the second ResNet50 as  $z_j = g(x_j; \theta_2)$ , where  $\theta_2$  denotes the parameters of the

second ResNet,  $g(x_j; \theta_2)$  is the non-linear mapping from the raw image  $x_j$  to binary-like code  $z_j$ . To encourage the *fch* layer to be binary codes, we utilize the hyperbolic tangent (tanh) function as the activation function. For the topmost layer, we define the binary code as :

$$b_i = \text{sign}(u_i) \quad (1)$$

$$h_j = \text{sign}(z_j). \quad (2)$$

Next, we will introduce the DAPH objective of joint learning the asymmetric hash functions and binary codes.

#### 3.3 Objective Formulation of DAPH

The key purpose of pairwise supervised hashing is to make Hamming distance between binary codes small (large) for similar (dissimilar) pairs. Many hashing loss functions have been devised by using above design principal. In particular, [22] propose a hinge-like loss function to learn compact binary codes. Other works [19, 25] adopt smooth  $L_2$  loss defined on the inner product between hash codes. In this work, inner product is utilized to be a good surrogate of the Hamming distance to quantify the pairwise similarity. Given the binary codes  $B = \{b_i\}_{i=1}^N \in \{-1, 1\}^{k \times N}$  and  $H = \{h_j\}_{j=1}^N \in \{-1, 1\}^{k \times N}$  for all images  $X$  from two asymmetric hashing mapping, the pairwise similarity preserving loss function is defined as follows:

$$L = \sum_{s_{ij} \in S} (\log(1 + e^{\Theta_{ij}}) - s_{ij}\Theta_{ij}) \quad (3)$$

where  $\Theta_{ij} = \frac{1}{2}b_i^T h_j$ . (3) is the negative log likelihood of pairwise similarity with the pairwise logistic function defined as follows:

$$p(s_{ij}|b_i, h_j) = \begin{cases} \sigma(\Theta_{ij}) & s_{ij} = 1 \\ 1 - \sigma(\Theta_{ij}) & s_{ij} = 0 \end{cases} \quad (4)$$

where  $\sigma(x) = \frac{1}{1+e^{-x}}$  is the sigmoid function. Furthermore, a good hashing method should produce binary codes with the properties: (1) **independence**, i.e., different bits in the binary codes are independent to each other; (2) **balance**, i.e. each bit hash a 50% chance of being 1 or -1. By incorporating the pairwise similarity-preserving loss and independent and balance constraint into the deep asymmetric hashing framework, we achieve the DPAH optimization problem:

$$\begin{aligned} \min_{B, H} L &= \sum_{s_{ij} \in S} (\log(1 + e^{\Theta_{ij}}) - s_{ij}\Theta_{ij}) \\ &+ \frac{\lambda}{2} \left( \left\| \frac{1}{N} BB^T - I \right\|^2 + \left\| \frac{1}{N} HH^T - I \right\|^2 \right) \\ &+ \frac{\beta}{2} \left( \|B\|_{N \times 1}^2 + \|H\|_{N \times 1}^2 \right) \end{aligned} \quad (5)$$

$$\text{s.t. } B, H \in \{-1, 1\}^{k \times N} \quad (6)$$

where  $I$  is the  $k \times k$  identity matrix and  $1_{k \times 1}$  is a vector which has  $k$  rows and all the elements equal to 1.

Since discrete optimization of (5) with binary constraint (6) is very challenging to solve, most existing methods apply continuous relaxation to the binary constraints. However, this continuous relaxation will give rise to the uncontrollable quantization error by binarizing continuous embedding to hash codes. To overcome

such limitation, in this work we utilize a novel strategy which can solve (5) in discrete way. To be specific, we replace the binary constraint  $b_i, h_j$  in (5) by the real-valued network outputs  $u_i$  and  $z_j$  respectively. To control the quantization error and close the gap between desired binary code and its relaxation, we impose two additional penalty terms on the  $u_i$  and  $z_j$  to approach the desired discrete binary codes  $b_i$  and  $h_j$  respectively. Then, we reformulate (5) as the following one:

$$\begin{aligned} \min_{B, H, U, Z} L &= \sum_{s_{ij} \in S} (\log(1 + e^{\Theta_{ij}}) - s_{ij} \Theta_{ij}) \\ &+ \frac{\alpha}{2} \left( \sum_{i=1}^n \|u_i - b_i\|^2 + \sum_{j=1}^n \|z_j - h_j\|^2 \right) \\ &+ \frac{\lambda}{2} \left( \left\| \frac{1}{N} U U^T - I \right\|^2 + \left\| \frac{1}{N} Z Z^T - I \right\|^2 \right) \\ &+ \frac{\beta}{2} (\|U\|_{N \times 1}^2 + \|Z\|_{N \times 1}^2) \quad (7) \\ \text{s.t. } B, H &\in \{-1, 1\}^{k \times N}. \quad (8) \end{aligned}$$

where  $U = \{u_i\}_{i=1}^N$ ,  $Z = \{z_j\}_{j=1}^N$  and  $\Theta_{ij} = \frac{1}{2} u_i^T z_j$ .

In our model, the binary code matrix for the training data  $X$  generated by the two asymmetric hash functions should be as close as possible. Hence, we add the regularization  $\|B - H\|^2$  in problem (7) as follows:

$$\begin{aligned} \min_{B, H, U, Z} L &= \sum_{s_{ij} \in S} (\log(1 + e^{\Theta_{ij}}) - s_{ij} \Theta_{ij}) \\ &+ \frac{\alpha}{2} (\|U - B\|^2 + \|Z - H\|^2) + \frac{\gamma}{2} (\|B - H\|^2) \\ &+ \frac{\lambda}{2} \left( \left\| \frac{1}{N} U U^T - I \right\|^2 + \left\| \frac{1}{N} Z Z^T - I \right\|^2 \right) \\ &+ \frac{\beta}{2} (\|U\|_{N \times 1}^2 + \|Z\|_{N \times 1}^2) \quad (9) \\ \text{s.t. } B, H &\in \{-1, 1\}^{k \times N}. \quad (10) \end{aligned}$$

### 3.4 Optimization

It is clear that the problem (9) is non-convex and non-smooth, which is in general an NP-hard problem due to the binary constraints  $B, H \in \{-1, 1\}^{k \times N}$ . To find a feasible solution, we propose an alternating optimization manner which is widely used in the hashing literature [25, 27]: updating one variables with others fixed. By taking advantage of the flexibility of the asymmetric hash function, we sequentially update the parameters of two deep neural networks and binary codes matrix  $B$  and  $H$  in the following alternating steps: **(i) Fix  $Z, B$  and update  $U$ .** When fixing  $B$  and the second network  $g(X; \theta_2)$ , the problem becomes to learn parameters of the first deep neural network  $\phi(X; \theta_1)$  by using stochastic gradient descent (SGD). In particular, in each iteration we sample a minibatch of images from the whole training dataset and use back-propagation algorithm to update the whole network. The derivatives of the loss

function are given by:

$$\begin{aligned} \frac{\partial L}{\partial U} &= \frac{1}{2} \sum_{s_{ij} \in S} (\sigma(\Theta_{ij}) - s_{ij}) Z + \alpha (U - B) \\ &+ \lambda \left( \frac{1}{N} U U^T - I \right) U + \beta U \mathbf{1}_{N \times 1}. \quad (11) \end{aligned}$$

we use the chain rule to compute  $\frac{\partial L}{\partial \theta_1}$  with  $\frac{\partial L}{\partial U}$  to update the parameters of neural network  $\theta_1$ .

**(i) Fix  $U, H$  and update  $Z$ .** In the same way, with  $B$  and  $U$  fixed, we can obtain the learned deep model by using SGD with a BP algorithm. In special, we compute the following gradient:

$$\frac{\partial L}{\partial Z} = \frac{1}{2} \sum_{s_{ij} \in S} (\sigma(\Theta_{ij}) - s_{ij}) U + \alpha (Z - H) \quad (12)$$

$$+ \lambda \left( \frac{1}{N} Z Z^T - I \right) Z + \beta Z \mathbf{1}_{N \times 1}. \quad (13)$$

**(i) Fix  $U, Z$  and update  $B, H$ .** When fixing  $U$  and  $R$ , we can rewrite (9) as

$$\begin{aligned} \min_{B, H} L_1 &= \alpha (\|U - B\|^2 + \|Z - H\|^2) + \gamma (\|B - H\|^2) \\ \text{s.t. } B, H &\in \{-1, 1\}^{k \times N} \quad (14) \end{aligned}$$

which is equal to

$$\max_{B, H} \text{Tr}(B^T Q) + \text{Tr}(H^T P) \quad (15)$$

$$\text{s.t. } B, H \in \{-1, 1\}^{k \times N} \quad (16)$$

where  $Q = \alpha U + \gamma H$  and  $P = \alpha Z + \gamma B$ . Finally we have the solution:

$$B = \text{sign}(Q) \quad (17)$$

$$H = \text{sign}(P). \quad (18)$$

We iteratively solve the above three sub-problems and the algorithm converges until it achieves the minimum value. Finally, two deep neural networks are produced for asymmetric hash functions. The proposed Deep Asymmetric Hashing algorithm is summarized in **Algorithm 1**.

## 4 EXPERIMENTS

In this section, we carry out extensive experiments to verify the efficiency of our method against several state-of-the-art hashing methods on three benchmark datasets.

### 4.1 Datasets

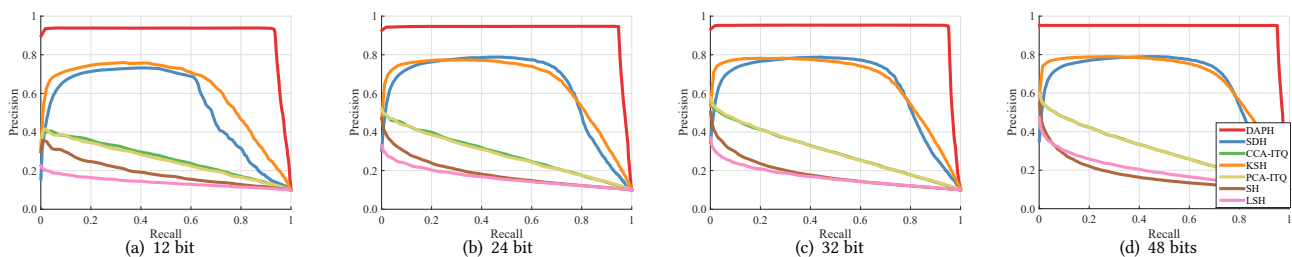
In the experiments, we choose to evaluate the effectiveness of the proposed methods on three datasets.

**CIFAR-10.** The CIFAR-10 [11] is a labeled subset of 80-million tiny images collection which consists of 60,000 32×32 color images in 10 classes with each of class 6,000 samples. The entire dataset is partitioned into two parts: a training set with 59,000 samples and a test set with 1000 samples.

**NUS-WIDE.** This dataset [2] contains nearly 270K images collected from Flickr. It is a multi-label dataset in which each image is annotated with one or multiple class labels from 81 concept tags. Following [33], we only use the images associated with the 21 most frequent concepts, where each of these concepts associates with at least 5,000 images.

**Table 1: Comparative results in MAP and mean precision of the top 500 retrieved neighbors (Precision@500) on CIFAR-10 with 12, 24, 32 and 48 bits.**

Method	MAP				Precision@500			
	12-bit	24-bit	32-bit	48-bit	12-bit	24-bit	32-bit	64-bit
LSH	15.42	18.11	18.23	21.06	20.37	27.73	29.35	35.07
SH	21.59	20.46	20.04	18.78	33.99	36.26	37.04	38.01
PCA-ITQ	29.12	31.47	33.42	33.72	39.95	47.01	50.52	51.41
KSH	66.49	73.24	74.21	75.32	68.19	74.15	76.53	77.42
CCA-ITQ	30.27	32.33	33.40	33.73	40.48	45.99	49.73	51.45
SDH	63.64	72.85	73.32	74.07	58.86	67.71	69.32	71.40
<b>DAPH</b>	<b>94.65</b>	<b>95.73</b>	<b>96.00</b>	<b>96.08</b>	<b>93.86</b>	<b>94.51</b>	<b>95.21</b>	<b>95.20</b>

**Figure 2: Precision-recall curves on CIFAR-10 with 12, 24, 32 and 48 bits, respectively.****Table 2: Comparison to deep hashing methods in MAP on CIFAR-10.**

Method	12-bit	24-bit	32-bit	48-bit
CNNH	48.4	47.6	47.2	48.9
DNNH	55.2	56.6	55.8	58.1
DHN	55.5	59.4	60.3	62.1
DAPH-S	50.12	57.03	71.69	81.42
<b>DAPH</b>	<b>75.69</b>	<b>82.13</b>	<b>83.07</b>	<b>84.48</b>

**MNIST.** The MNIST dataset consists of 70,000 images of hand-written digits from ‘0’ to ‘9’. The training set contains 60,000 images, the query set contains 10,000 images.

In CIFAR-10, we randomly select 1K query images (100 images per object class) as the test query set. For conventional hand-crafted hashing methods, we use the rest images as training dataset; for deep hashing methods, we randomly select 500 images per class as the training set following the experimental protocols [16] for fair comparison. The pairwise similarity matrix  $S$  is constructed based on the class labels (i.e., the value corresponding the image pair from the same class is set to one and zero otherwise). In MNIST, we randomly select 10K as the query set and the other 60K as the training samples. The similarity matrix is constructed based on the category labels as well. In NUSWIDE, we randomly select 500 images per class as the training set and 100 images per class to form the test query set. The similarity pairs for training are

randomly constructed using image labels: each pair is considered similar (dissimilar) if they share at least one (none) semantic label. For each dataset, we use the training set to learn the network parameters and use the query set to test its effectiveness.

## 4.2 Evaluation Protocol and Baselines

We use the following evaluation metric to measure the performance: hashing ranking performance means average precision (MAP), mean precision of the top 500 retrieval samples (Precision@500). We also show the precision-recall curves of three datasets. For fair comparison, all of the methods use identical training and test datasets.

We compare our method with several state-of-the-art hashing algorithms. These methods can be categorized into three classes: three unsupervised methods including Locality-sensitive Hashing (LSH) [6], SH [32], PCA-ITQ [8], supervised hashing methods including KSH [19], SDH [27], CCA-ITQ [8] and deep hashing methods CNNH [33], DNNH [14], DHN [16]. We use the available codes and suggested parameters in the original papers of these approaches. For fair comparison, we evaluate these hand-crafted hashing methods LSH, SH, ITQ, KSH and SDH on the feature extracted from the last hidden layer of the ResNet50 model pre-trained on the ImageNet dataset [3]. Specifically, we re-arrange the neuron responses on the layer right below the hash layer into vector formats and feed them into baselines. We find that using ResNet50 features greatly improves the performance of these hashing methods compared with hand-crafted features which will be shown in the following experiments. For deep hashing methods, we first resize all the images

**Algorithm 1** Deep Asymmetric Pairwise Hashing (DAPH)

---

**Input:** Training data  $X$ ; similarity matrix  $S$ ; binary code length  $k$ ; parameters  $\alpha, \gamma, \lambda$  and  $\beta$ ; iteration number  $T$ .  
**Output:** Parameters of  $\theta_1$  and  $\theta_2$  of the deep network and binary code matrix  $B, H$  of training set;

- 1: Initialize  $B, H \in \{-1, 1\}^{k \times N}$ ; Initialize neural network parameters of  $\theta_1$  and  $\theta_2$ ;
- 2: **repeat**
- 3:   update  $\theta_1$  and  $\theta_2$  by minimizing the image classification error;
- 4: **until** converge
- 5: **repeat**
- 6:   **for**  $i = 1 \rightarrow T$  **do**
- 7:     Forward computation to compute  $u_i = \phi(x_i; \theta_1)$  from the raw images in mini-batch;
- 8:     Compute derivation according to (11);
- 9:     Update the neural network  $\theta_1$  by utilizing back propagation.
- 10:   **end for**
- 11:   **for**  $i = 1 \rightarrow T$  **do**
- 12:     Forward computation to compute  $z_j = \phi(x_j; \theta_2)$  from the raw images in mini-batch;
- 13:     Compute derivation according to (12);
- 14:     Update the neural network  $\theta_2$  by utilizing back propagation.
- 15:   **end for**
- 16:   Update  $B$  according to (17);
- 17:   Update  $H$  according to (18);
- 18: **until** converge or reach maximum iterations;

---

to be 224×224 pixels and then directly use the raw image pixels as inputs.

Our model is implemented with open-source Keras framework [1]. Training is done on a standard desktop with a GeForce GTX TITAN X with 12GB memory. We employ the deep residual network (ResNet50) architecture, and initialize our ResNet50 using the pre-trained model and fine-tune the convolutional layers and fully-connected layers on the corresponding training set. During training, we use stochastic gradient descent with momentum to 0.9 and weight decay to 0.0001. The initial learning rate is 0.0001. We fix the batch as 64. After the algorithm converged, we find that the two deep ResNet50 can achieve almost similar retrieval performance in CIFAR10 dataset. So in all the experiments, we choose the first ResNet50 as the final hash function uniformly. The parameters  $\alpha, \gamma, \beta$  and  $\lambda$  are empirically set as 10, 10, 0.01 and 0.01 respectively in all the experiments.

### 4.3 Results and Discussion

**Result on CIFAR-10.** We first report the performance of our model on CIFAR-10. Table 1 and 2 illustrates the score of MAP, precision@500 of compared methods using various numbers of bits. Figure 2 displays the precision-recall curves for all algorithms with 12, 24, 32 and 48 hash bits. We can see that the proposed method outperforms all of the other methods in all cases from Table 1. In terms of MAP, the proposed method achieves substantially

better performance at all code lengths. The MAP of our method consistently outperforms SDH, which is the current state-of-the-art supervised hashing method. In particular, compared to SDH, the MAP results of DAPH indicate a relative increase of 20.46%, 18.73%, 20.9%, 17.29% respectively. The substantial superior performance of DAPH verifies the benefit of joint learning feature representations and hash codes rather than the traditional cascaded scheme (i.e. feature extraction followed by hashing).

The Precision@500 of our method consistently outperforms compared methods at all code lengths as we can see from Table 1. Figure 2 shows the precision-recall curves for different lengths of hash bits. The results with 12, 24, 32 and 48 bits confirm the performance gains of DAPH in Table 1.

Among the deep hashing methods, please note the MAP results of CNNH, DNNH and DHN are from [33], [14], [43]. The MAP results with different code lengths are listed in Table 2. It's observed that our DAPH yields the highest MAP score in all cases. For example, our method obtains 75.69%, 82.13%, 83.07%, 84.48% which are higher than DHN by 20.19%, 22.73%, 22.77%, 22.38% respectively. The underlying reason why our method exceeds DHN method by a large margin may be that by using two deep networks to construct distinct asymmetric hash functions, our methods can capture more information and has higher learning capability and is able to exploit more semantic information than symmetric hashing setting. We also investigate that one variant of DAPH: DAPH-s, which is the DAPH variant without asymmetric deep hash functions. We can observe that without using asymmetric hash functions, DAPH-S suffers from large MAP decreases. These results validate that the proposed deep asymmetric pairwise hashing method can effectively exploit the pairwise affinity and lead to remarkable hash code.

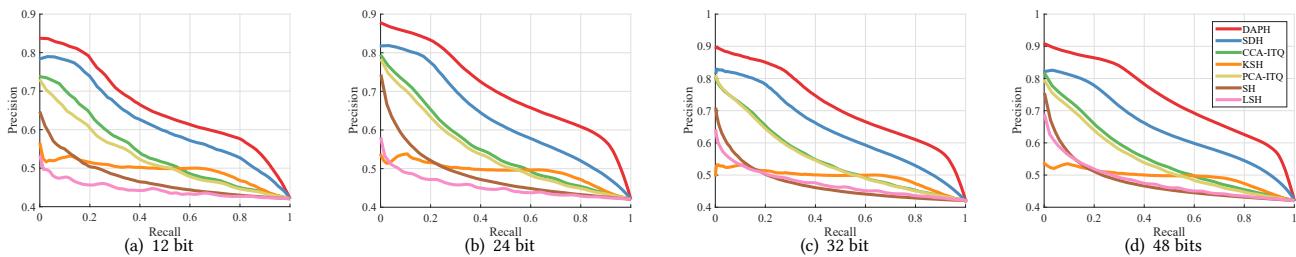
**Result on NUSWIDE.** The MAP results for our method and other traditional baselines with ResNet50 features on NUSWIDE are reported in Table 3 and Table 4. We can find that our method can outperform all the other baselines and all the deep hashing methods in term of MAP. For conventional hashing algorithm, SDH achieves the best accuracies both of MAP and Precision@500. In general, those deep hashing methods outperform the conventional hash learning methods in most cases. Among the deep hashing methods, we can observe that our DAPH yields the highest MAP in all cases. This may be explained that CNNH is a two stage deep methods which trains a deep network to fit the binary codes computed from the pairwise similarity matrix; DNNH uses a fixed-margin loss and piecewise-linear activation function to train deep networks which may cause information loss and objective oscillations in back propagation.

Compared to the state-of-the-art deep hashing method DHN, we achieve absolute superiority 0.87%, 3.56%, 3.9%, 5.84% of in average MAP for different bits on the NUSWIDE. The improvement is more clear at high code length, which demonstrates the advantage of the asymmetric hashing model for image retrieval again.

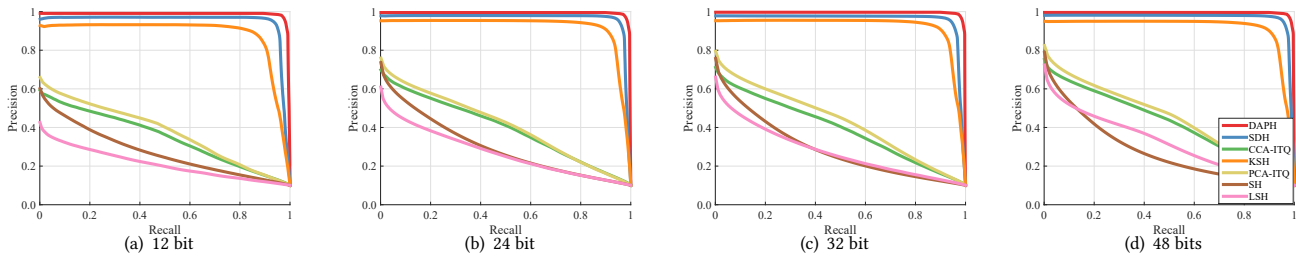
With respect to Precision@500, the proposed methods show superior performance gains against all the conventional hashing methods. For example, compared to the corresponding second best method SDH, the proposed method shows a relative increase of 3.75%, 4.64%, 5.71%, 6.93% respectively. Figure 3 represents the

**Table 3: Comparative results in MAP and mean precision of the top 500 retrieved neighbors (Precision@500) on NUSWIDE with 12, 24, 32 and 48 bits.**

Method	MAP				Precision@500			
	12-bit	24-bit	32-bit	48-bit	12-bit	24-bit	32-bit	64-bit
LSH	53.03	53.07	55.58	54.27	43.56	44.22	45.74	45.62
SH	53.25	52.41	54.29	53.56	55.48	56.14	58.39	57.74
PCA-ITQ	59.60	60.13	60.22	60.58	64.92	69.03	69.64	70.35
KSH	53.00	52.78	53.07	53.13	52.07	53.02	52.55	49.25
CCA-ITQ	61.88	61.60	62.68	62.28	71.46	72.20	73.24	73.14
SDH	64.59	63.94	60.33	62.71	77.36	79.86	80.20	80.07
<b>DAPH</b>	<b>71.67</b>	<b>77.06</b>	<b>78.70</b>	<b>81.64</b>	<b>81.16</b>	<b>84.50</b>	<b>85.91</b>	<b>87.00</b>



**Figure 3: Precision-recall curves on NUSWIDE with 12, 24, 32 and 48 bits, respectively.**



**Figure 4: Precision-recall curves on MNIST with 12, 24, 32 and 48 bits, respectively.**

**Table 4: Comparison to deep hashing methods in MAP on NUSWIDE.**

Method	12-bit	24-bit	32-bit	48-bit
CNNH	62.3	63.0	62.9	62.5
DNNH	67.4	69.7	71.3	71.5
DHN	70.8	73.5	74.8	75.8
<b>DAPH</b>	<b>71.67</b>	<b>77.06</b>	<b>78.70</b>	<b>81.64</b>

precision-recall curves for our method and compared conventional methods, which corresponds to the trend in Table 3.

**Result on MNIST.** Table 6 shows comparative MAP and Precision@500 respectively. In term of MAP, our method achieves nearly

**Table 5: Evaluation of our method with/without the constraint of balance and independence on the image retrieval task. -: This operation is not applied;  $\checkmark$ : Applied. The results are reported in MAP and Precision@500 on CIFAR-10.**

Balance	Independence	MAP		Precision@500	
		24-bit	32-bit	24-bit	32-bit
-	-	78.38	76.56	80.08	82.04
-	$\checkmark$	78.74	80.03	80.18	82.08
$\checkmark$	-	78.63	79.70	80.08	81.88
$\checkmark$	$\checkmark$	<b>82.13</b>	<b>83.07</b>	<b>81.37</b>	<b>84.33</b>



**Table 6: Results in MAP and Precision@500 of the compared methods on MNIST .**

Method	MAP				Precision@500			
	12-bit	24-bit	32-bit	48-bit	12-bit	24-bit	32-bit	64-bit
LSH	24.51	30.80	30.71	37.20	36.45	49.23	52.60	57.31
SH	30.32	32.81	31.25	29.93	51.37	59.68	60.41	61.46
PCA-ITQ	43.31	46.02	47.91	49.29	59.76	66.45	68.94	71.12
KSH	91.48	92.48	93.93	93.78	92.98	93.69	94.74	94.82
CCA-ITQ	40.31	44.13	44.02	46.55	55.48	63.21	63.45	67.12
SDH	96.48	97.71	97.21	97.90	97.04	97.92	97.80	98.13
DAPH	<b>99.26</b>	<b>99.61</b>	<b>99.71</b>	<b>99.64</b>	<b>99.07</b>	<b>99.52</b>	<b>99.67</b>	<b>99.55</b>

perfect scores of 99.26%, 99.61%, 99.71%, 99.64%, thus significantly outperforms other hashing methods at all code lengths. Among the compared methods, the data-independence LSH improves the performance as the code size increases, and it even surpasses the SH at code length is 48 bits. This behavior may due to the theoretic convergence guarantee of LSH with long hash codes.

Table 6 also gives the Precision@500 results with different code lengths and our DAPH still works the best. Among other methods, SDH achieves best results, which implies the directly learning the binary codes without relaxation is preferable than the continuous solution. However, our method DAPH has a consistent advantage over the SDH which demonstrates the power of the asymmetric deep hashing method again. Figure 4 displays the precision-recall curves for all code lengths (from 12-bit to 48-bit) on the dataset MNIST, which confirms the performance gains of our method in Table 6.

**The Impact of Bits Independence and Balance.** In this part, we validate the effectiveness of the two well-known properties of binary codes: bits independence and balance. The performance of our method with or without each of the properties is shown in Table 5. The database of CIFAR-10 is used for evaluation. As we can see, the properties play important roles in binary code learning. In particular, our method obtains better results by incorporating both these properties in binary code than discarding them or keep only one in all cases in both MAP, Precision@500.

## 5 CONCLUSION

In this paper, we proposed a novel supervised deep hashing approach called deep asymmetric pairwise hashing (DAPH), which integrated feature learning and asymmetric hash function learning into the end-to-end deep learning framework. DAPH was designed to jointly learn pairwise similarity-preserving binary codes and semantic information rich hash functions. By taking advantage of the asymmetric hashing scheme, an efficient alternating optimization algorithm was proposed to solve the discrete code optimization problem. Our comprehensive evaluations on three image benchmarks showed that DAPH outperformed many recent supervised hashing algorithms, which validated the effectiveness of the deep asymmetric hashing method.

As well as image search, binary hashing has recently been applied to boost many other tasks, such as classification [26], clustering [34, 38], recommendation systems [41] and action recognition [23]. The proposed DAPH method is also supposed to improve these problems due to its advantages. Besides, applying the asymmetric method of this work into the recent vision-language tasks (*e.g.*, video captioning [40, 42]) would be an interesting problem.

## ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Project 61502081, Project 61572108 and Project 61632007, in part by the Fundamental Research Funds for the Central Universities under Project ZYGX2014Z007.

## REFERENCES

- [1] Francois Chollet. 2015. Keras. <https://github.com/fchollet/keras>. (2015).
- [2] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao Zheng. 2009. NUS-WIDE: A Real-World Web Image Database from National University of Singapore. In *Proc. of ACM Conf. on Image and Video Retrieval*.
- [3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*. 248–255.
- [4] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. 2014. DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. In *Proc. ICML*. 647–655.
- [5] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. 2015. Deep Hashing for Compact Binary Codes Learning. In *Proc. CVPR*. 2475–2483.
- [6] Aristides Gionis, Piotr Indyk, and Rameez Motwani. 1999. Similarity search in high dimensions via hashing. In *Proc. VLDB*. 518–529.
- [7] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*. 580–587.
- [8] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 12 (2013), 2916–2929.
- [9] Qing-Yuan Jiang and Wu-Jun Li. 2017. Deep Cross-Modal Hashing. In *Proc. CVPR*. 3232–3240.
- [10] Weihao Kong and Wu-Jun Li. 2012. Isotropic hashing. In *Proc. NIPS*. 1655–1663.
- [11] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. *Tech. Rep.* (2009).
- [12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*. 1097–1105.
- [13] Brian Kulis and Trevor Darrell. 2009. Learning to hash with binary reconstructive embeddings. In *Proc. NIPS*. 1042–1050.
- [14] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *Proc. CVPR*. 3270–3278.
- [15] Guosheng Lin, Chunhua Shen, and Anton van den Hengel. 2015. Supervised hashing using graph cuts and boosted decision trees. *IEEE Trans. Pattern Anal. Mach. Intell.* 37, 11 (2015), 2317–2331.
- [16] Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. 2016. Deep supervised hashing for fast image retrieval. In *Proc. CVPR*. 2064–2072.



- [17] Li Liu, Ling Shao, Fumin Shen, and Mengyang Yu. 2017. Discretely Coding Semantic Rank Orders for Image Hashing. In *Proc. CVPR*. 1425–1434.
- [18] Li Liu, Fumin Shen, Yuming Shen, Xianglong Liu, and Ling Shao. 2017. Deep Sketch Hashing: Fast Free-hand Sketch-Based Image Retrieval. In *Proc. CVPR*. 2862–2871.
- [19] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *Proc. CVPR*. 2074–2081.
- [20] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2011. Hashing with graphs. In *Proc. ICML*. 1–8.
- [21] Behnam Neyshabur, Nati Srebro, Ruslan R Salakhutdinov, Yury Makarychev, and Payman Yadollahpour. 2013. The power of asymmetry in binary hashing. In *Proc. NIPS*. 2823–2831.
- [22] Mohammad Norouzi and David M Blei. 2011. Minimal loss hashing for compact binary codes. In *Proc. ICML*. 353–360.
- [23] Jie Qin, Li Liu, Ling Shao, Bingbing Ni, Chen Chen, Fumin Shen, and Yunhong Wang. 2017. Binary Coding for Partial Action Analysis with Limited Observation Ratios. In *Proc. CVPR*. 146–155.
- [24] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.
- [25] Fumin Shen, Wei Liu, Shaoting Zhang, Yang Yang, and Heng Tao Shen. 2015. Learning binary codes for maximum inner product search. In *Proc. ICCV*. 4148–4156.
- [26] Fumin Shen, Yadong Mu, Yang Yang, Wei Liu, Li Liu, Jingkuan Song, and Heng Tao Shen. 2017. Classification by Retrieval: Binarizing Data and Classifier. In *Proc. SIGIR*.
- [27] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised Discrete Hashing. In *Proc. CVPR*. 37–45.
- [28] Fumin Shen, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, and Zhenmin Tang. 2013. Inductive Hashing on Manifolds. In *Proc. CVPR*. 1562–1569.
- [29] Christoph Strecha, Alex Bronstein, Michael Bronstein, and Pascal Fua. 2012. LDAHash: Improved matching with smaller descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.* 34, 1 (2012), 66–78.
- [30] Yi Sun, Yuheng Chen, Xiaogang Wang, and Xiaoou Tang. 2014. Deep learning face representation by joint identification-verification. In *Proc. NIPS*. 1988–1996.
- [31] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. 2017. A survey on learning to hash. *IEEE Trans. Pattern Anal. Mach. Intell.* PP (2017).
- [32] Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. In *Proc. NIPS*. 1753–1760.
- [33] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. Supervised Hashing for Image Retrieval via Image Representation Learning. In *Proc. AAAI*. 2156f?–2162.
- [34] Ivor Tsang Fumin Shen Xiaobo Shen, Weiwei Liu and Quan-Sen Sun. 2017. Compressed K-means for Large-scale Clustering. In *Proc. AAAI*. 2527–2533.
- [35] Liang Xie, Jialie Shen, and Lei Zhu. 2016. Online Cross-Modal Hashing for Web Image Retrieval. In *Proc. AAAI*. 294–300.
- [36] H. F. Yang, K. Lin, and C. S. Chen. 2017. Supervised Learning of Semantics-Preserving Hash via Deep Convolutional Neural Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PP (2017).
- [37] Yang Yang, Yadan Luo, Weilun Chen, Fumin Shen, Jie Shao, and Heng Tao Shen. 2016. Zero-Shot Hashing via Transferring Supervised Knowledge. In *Proc. MM*. 1286–1295.
- [38] Yang Yang, Fumin Shen, Zi Huang, and Heng Tao Shen. 2016. A Unified Framework for Discrete Spectral Clustering. In *Proc. IJCAI*. 2273–2279.
- [39] Yang Yang, Fumin Shen, Heng Tao Shen, Hanxi Li, and Xuelong Li. 2015. Robust discrete spectral hashing for large-scale image semantic indexing. *IEEE Trans. Big Data* 1, 4 (2015), 162–171.
- [40] Yimeng Li Long Chen Jun Xiao Yunan Ye, Zhou Zhao and Yueting Zhuang. 2017. Video Question Answering via Attributed-Augmented Attention Network Learning. In *Proc. SIGIR*.
- [41] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete Collaborative Filtering. In *Proc. SIGIR*. 325–334.
- [42] Zhou Zhao, Qifan Yang, Deng Cai, Xiaofei He, and Yueting Zhuang. 2017. Video Question Answering via Hierarchical Spatio-Temporal Attention Networks. In *Proc. IJCAI*.
- [43] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. 2016. Deep Hashing Network for Efficient Similarity Retrieval. In *Proc. AAAI*. 2415–2421.
- [44] Lei Zhu, Jialie She, Xiaobai Liu, Liang Xie, and Liqiang Nie. 2016. Learning Compact Visual Representation with Canonical Views for Robust Mobile Landmark Search. In *Proc. IJCAI*. 3959–3965.
- [45] Lei Zhu, Jialie Shen, Liang Xie, and Zhiyong Cheng. 2017. Unsupervised Visual Hashing with Semantic Assistant for Content-Based Image Retrieval. *IEEE Trans. on Knowl. and Data Eng.* 29, 2 (2017), 472–486.