The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

# Supervised Deep Hashing for Hierarchical Labeled Data

**Dan Wang,[1] Heyan Huang,[1]* Chi Lu,[1] Bo-Si Feng,[1]**
**Guihua Wen,[2] Liqiang Nie,[3] Xian-Ling Mao[1]**
[1]Beijing Institute of Technology, China
[2]South China University of Technology, China
[3]Shandong University, China
{wangdan12856, hhy63, luchi, maoxl}@bit.edu.cn, fengbosi1994@163.com
crghwen@scut.edu.cn, nieliqiang@gmail.com

## Abstract

Recently, hashing methods have been widely used in large-scale image retrieval. However, most existing supervised hashing methods do not consider the hierarchical relation of labels, which means that they ignored the rich semantic information stored in the hierarchy. Moreover, most of previous works treat each bit in a hash code equally, which does not meet the scenario of hierarchical labeled data. To tackle the aforementioned problems, in this paper, we propose a novel deep hashing method, called supervised hierarchical deep hashing (SHDH), to perform hash code learning for hierarchical labeled data. Specifically, we define a novel similarity formula for hierarchical labeled data by weighting each level, and design a deep neural network to obtain a hash code for each data point. Extensive experiments on two real-world public datasets show that the proposed method outperforms the state-of-the-art baselines in the image retrieval task.

## Introduction

Nearest neighbour (NN) search plays a pivotal role in multimedia and related areas, such as image retrieval, pattern recognition, and computer vision (Zheng et al. 2015). In big data applications, it is, however, time-consuming to return the exact nearest neighbours to the given queries. Hence, approximate nearest neighbour (ANN) search algorithms with improved speed and memory saving have drawn more and more attention from researchers in this big data era (Andoni and Indyk 2008).

Due to its fast retrieval speed and low storage cost, similarity-preserving hashing has been widely used for ANN search (Zhu et al. 2016). The central idea of hashing is to map the data points from the original feature space into binary codes in the Hamming space and preserve the pairwise similarities in the original space. With the binary-code representation, hashing enables constant or sub-linear time complexity for ANN search (Gong and Lazebnik 2011; Zhang et al. 2014). Moreover, hashing can reduce the storage cost dramatically.

Compared with traditional data-independent hashing methods like Locality Sensitive Hashing (LSH) (Gionis, Indyk, and Motwani 1999; Raginsky and Lazebnik 2009) which do not use any data for training, data-dependent hashing methods, can achieve better accuracy with shorter codes by learning hash functions from training data (Gong and Lazebnik 2011; Zhang et al. 2014). Existing data-dependent methods can be further divided into three categories: unsupervised methods (He, Wen, and Sun 2013; Gong and Lazebnik 2011; Shen et al. 2015; Zhu et al. 2017), supervised methods (Liu et al. 2012; Zhang et al. 2014; Yuan et al. 2017), and semi-supervised methods (Wang, Kumar, and Chang 2010; Zhang, Peng, and Zhang 2016; Zhang and Zheng 2017). Unsupervised hashing works by preserving the Euclidean similarity between the attributes of training points, while supervised and semi-supervised hashing try to preserve the semantic similarity constructed from the semantic labels of the training points (Zhang et al. 2014; Kang, Li, and Zhou 2016). Although there are also some works to exploit other types of supervised information like the ranking information for hashing (Li et al. 2013), the semantic information is usually given in the form of pairwise labels indicating whether two data points are known to be similar or dissimilar. As far as we know, these supervised and semi-supervised methods can mainly be used to deal with the data with non-hierarchical labels.

However, there are indeed lots of hierarchical labeled data, such as Imagenet (Deng et al. 2009), IAPRTC-12(Escalante et al. 2010) and CIFAR-100(Krizhevsky 2009). Intuitively, we can simply take hierarchical labeled data as non-hierarchical labeled data, and then take advantage of the existing algorithms. Obviously, it cannot achieve optimal performance, because most of the existing methods are essentially designed to deal with non-hierarchical labeled data which do not consider special characteristics of hierarchical labeled data. For example, in Figure 1, if taking the hierarchical ones as non-hierarchical labeled data, images $I_a$ and $I_b$ have the same label “Rose”, the label of the image $I_c$ is “Sunflower”, and the labels for $I_d$ and $I_e$ are respectively “Ock” and “Tiger”. Given a query $I_q$ with the ground truth label “Rose”, the retrieved results may be “$I_a$, $I_b$, $I_e$, $I_d$, and $I_c$” in descending order without considering the hierarchy. It does not make sense that the ranking positions of images $I_e$ and $I_d$ are higher than that of $I_c$, because the image $I_c$ is also

*Corresponding Author

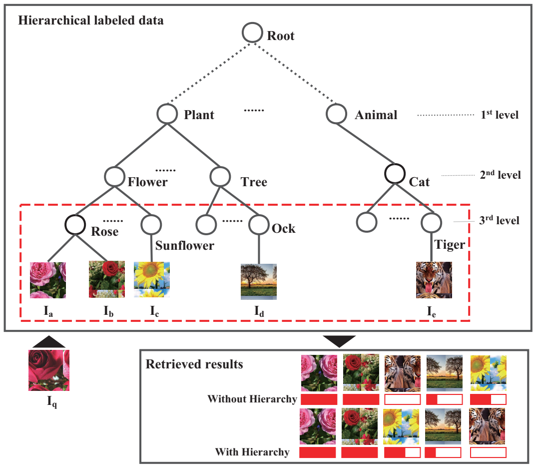


Figure 1: A hierarchical labeled dataset. The height of the hierarchy is four. The different retrieved results where whether the hierarchical relation is considered are demonstrated. The longer a red bar is, the more similar the corresponding image is.

a flower although it is not a rose.

To address the aforementioned problems, we propose a novel supervised hierarchical deep hashing method for hierarchical labeled data, denoted as SHDH. Specifically, we define a novel similarity formula for hierarchical labeled data by weighting each level, and design a deep neural network to obtain a hash code for each data point. Extensive experiments on two real-world public datasets show that the proposed method outperforms the state-of-the-art baselines in the image retrieval task.

The main contributions in this paper are:

- To the best of our knowledge, the proposed SHDH is the first method to utilize the hierarchical relation in hierarchical labeled data. Meanwhile, the bits in the hash code learnt by SHDH are weighted.
- We design a novel similarity formula for hierarchical labeled data by weighting each level.
- Experimental results on two public datasets demonstrate that the proposed SHDH method is effective.

The rest of this paper is organized as follows. Section 2 introduces some related works. Section 3 describes the details of the proposed model and the parameter learning algorithm. Experimental results and analysis are demonstrated in Section 4. Finally, Section 5 presents some concluding remarks.

## Related Work

The existing hashing methods can be grouped into two categories: data-independent and data-dependent methods. For data-independent methods, the hashing functions are learned without any training data. Representative data-independent methods include LSH (Andoni and Indyk 2008; Gionis, Indyk, and Motwani 1999), Shift-invariant Kernels Hashing (SIKH) (Raginsky and Lazebnik 2009), and lots of extensions (Kulis, Jain, and Grauman 2009; Datar et al. 2004). For data-dependent methods, their hashing functions are learned from training data. Generally speaking, data-independent methods often require more number of bits than data-dependent methods to achieve satisfactory performance.

Furthermore, data-dependent hashing methods can be divided into three folds: unsupervised, supervised, and semi-supervised methods. Unsupervised methods use unlabeled data to learn hash functions and try to keep the neighborhood relation of data in the original space. Representative unsupervised hashing methods include K-means Hashing (KMH) (He, Wen, and Sun 2013), Iterative Quantization (ITQ) (Gong and Lazebnik 2011), Asymmetric Innerproduct Binary Coding (AIBC) (Shen et al. 2015), and Semantic-assisted Visual Hashing (SAVH) (Zhu et al. 2017). Usually, the hashing methods with supervised semantic information outperform unsupervised methods due to the semantic gap problem. In supervised hashing approaches, label information is utilized to build the similarity matrix of training data to learn the set of hashing functions. Lots of algorithms in this category have been proposed, including Kernel-based Supervised Hashing (KSH) (Liu et al. 2012), Latent Factor Hashing (LFH) (Zhang et al. 2014), Column Sampling Based Discrete Supervised Hashing (COSDISH) (Kang, Li, and Zhou 2016), and Reconstruction-based Supervised Hashing (RSH) (Yuan et al. 2017). Semi-supervised algorithms (Zhang, Peng, and Zhang 2016; Zhang and Zheng 2017) use both the labeled samples and the unlabeled ones to learn the hash code. For example, the Semi-supervised Hashing (SSH) (Wang, Kumar, and Chang 2010) minimizes the error between the pairwise labeled data and maximizes the variance of hash code over the labeled and unlabeled data.

Despite their significance, few existing efforts in hashing area focus on hierarchical labeled data. Meanwhile, some recent methods performing simultaneously feature learning and hash code learning with deep neural networks, have shown better performance (Li, Wang, and Kang 2016; Qiu et al. 2017). Thus, we will present an effective deep learning approach for hierarchical labeled data to perform simultaneously feature learning and hash code learning.

## Method

### Hierarchical Similarity

As shown in Figure 1, it can be found that the "Root" node is the ancestor of all data points, thus it has no discriminative ability, and will not be considered in the definition of hierarchical similarity. Moreover, in this paper, different with the definition of the level in the tree data structure, the level of a node is defined by the number of edges from the node to the root node. Thus, the levels of a hierarchy from top to down can be denoted as $1^{st}$ level, $2^{nd}$ level, and so on. For example, in Figure 1, "Plant" node is located in the $1^{st}$ level.

It is reasonable that images have distinct similarity in different levels in a hierarchy. For example, in Figure 1, images $I_a$ and $I_c$ are similar in the second level because they are both flower. However, they are dissimilar in the third level because $I_a$ belongs to rose but $I_c$ belongs to sunflower. In the light of this, we have to define hierarchical similarity for two images in hierarchical labeled data. To define the hierarchical similarity, we first introduce two definitions: level similarity and level weight.

**Definition 1 (Level Similarity)** *For two images i and j, the similarity at the $k^{th}$ level in a hierarchy is defined as:*

$$s_{ij}^k = \begin{cases} 1, & \text{if } Ancestor_k(i) = Ancestor_k(j); \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

*where $Ancestor_k(i)$ is the ancestor node of image $i$ at the $k^{th}$ level.*

Equation (1) means that if images $i$ and $j$ share the common ancestor node in the $k^{th}$ level, they are similar at this level. On the contrary, they are dissimilar. For example, in Figure 1, the level similarities between images $I_a$ and $I_c$ at different levels are: $s^1_{I_aI_c} = 1$, $s^2_{I_aI_c} = 1$, and $s^3_{I_aI_c} = 0$.

Intuitively, the higher level is more important, because we cannot reach the right descendant nodes if we choose a wrong ancestor. We thus have to consider the weight for each level in a hierarchy. Any functions which satisfy the following two conditions could be used as the level weight: (1) $u_k > u_{k+1}$, where $k \in [1, 2, \cdots, K-1]$. It satisfies the demand where the influence of ancestor nodes is greater than that of descendant nodes. (2) $\sum_{k=1}^{K} u_k = 1$. Thus, we define level weight as:

**Definition 2 (Level Weight)** *The importance of $k^{th}$ level in a hierarchy where the number of levels is K, can be estimated as:*

$$u_k = \frac{2(K+1-k)}{K(K+1)}, \tag{2}$$

*where $k \in [1, 2, \cdots, K]$.*

Based upon the two definitions above, the final hierarchical similarity between images $i$ and $j$ can be calculated by the following definition:

**Definition 3 (Hierarchical Similarity)** *For two images i and j in a hierarchy where the number of levels is K, their hierarchical similarity is:*

$$s_{ij} = 2\sum_{k=1}^{K} u_k s_{ij}^k - 1. \tag{3}$$

Equation (3) scales the final hierarchical similarity into a real value between -1 and 1. It guarantees that the more common hierarchical labels image pairs have, the more similar they are.

## Supervised Hierarchical Deep Hashing

Figure 2 shows the proposed deep learning architecture. The proposed SHDH model consists of two parts: feature learning and hash function learning. The feature learning part includes a convolutional neural network (CNN) component and two fully-connected layers. The CNN component contains five convolutional layers. Specifically, the first convolutional layer filters the input images with 64 kernels of size 11 × 11 with a stride of four pixels. The output of the first convolutional layer will be response-normalized and max-pooled (size 2 × 2) to be the input of the second convolutional layer. The second layer has 256 kernels of size 5 × 5 with a stride of one pixel and a pad of size 2 pixels. Its output will be response-normalized and max-pooled (size 2 × 2) to be the input of the third convolutional layer. The third, fourth and fifth layers have 256 kernels of size 3 × 3 with a stride of one pixel and a pad of size one pixel. The fifth layer has a max-pooling layer with filter of size 2 × 2. After the CNN component, the architecture holds two fully-connected layers which have 4,096 hidden units. The activation function used in this part is Rectified Linear Units (ReLu) (Krizhevsky, Sutskever, and Hinton 2012).

The hash function learning part includes a hashing layer and an independent weighting layer. The hashing functions are learnt by the hashing layer whose size is the length of hash code. And no activation function used in this layer. Note that the hashing layer is divided into $K$-segments and $K$ is the number of levels in a hierarchy. The size of $1^{st} \sim (K-1)^{th}$ segments is $\lfloor \frac{L}{K} \rfloor$, where L is the length of hash code. And the size of the last segment is $L - \lfloor \frac{L}{K} \rfloor \times (K-1)$. $L_k$ is used to represent the size of $k^{th}$ segment, where $k \in [1, 2, \cdots, K]$. Here, there is an implicit assumption that $L$ is larger than $K$. It is a reasonable assumption, since the height of hierarchical labeled data is usually small, while the length of hash code is usually large, such as 16, 32, and 128 bits. Besides, the values in the weighting layer are the weights calculated by Eq. (2) from the hierarchical labeled data, which are used to adjust the Hamming distance among segmented hash code. Each value in the weighting layer weights a corresponding segment in the hashing layer.

**Objective Function** Given a hierarchical labeled dataset $\boldsymbol{X} = \{\boldsymbol{x_i}\}_{i=1}^{N}$ where $\boldsymbol{x_i}$ is the $i^{th}$ data point and $N$ is the number of data points. Its semantic matrix $\boldsymbol{S} = \{s_{ij}\}$ can be built via Eq. (3), where $s_{ij} \in [-1, 1]$. The goal of our SHDH is to learn a $L$-bit binary codes vector $\boldsymbol{h_i} \in \{-1, 1\}^L$ for each point $\boldsymbol{x_i}$, $L$ is the length of hash code.

Assume there are $M+1$ layers in our deep network containing $M-1$ layers for feature learning, one hashing layer and one weight layer. In Figure 2, $M = 8$. The output of the whole network is: $\boldsymbol{b_i} = \boldsymbol{W}^T \boldsymbol{f}(\boldsymbol{x_i}; \boldsymbol{\theta}) + \mathbf{v} \in \mathbb{R}^L$, where the mapping $\boldsymbol{f} : \mathbb{R}^d \rightarrow \mathbb{R}^{4,096}$ is parameterized by $\boldsymbol{\theta}$ and $\boldsymbol{\theta}$ represents the parameters of the feature learning part. $\boldsymbol{W} \in \mathbb{R}^{4,096 \times L}$ is the projection matrix to be learnt at the $M^{th}$ layer of the network, $\boldsymbol{v} \in \mathbb{R}^L$ is the bias.

Now, we can perform hashing for the output $\boldsymbol{b_i}$ at the top layer of the network to obtain binary codes as follows: $\boldsymbol{h_i} = sgn(\boldsymbol{b_i})$. The procedure above is forward. To learn the parameters of our network, we have to define an objective function.

First, for an image $\boldsymbol{x_i}$, its hash code is $\boldsymbol{h_i} \in \{-1, 1\}^L$ consisting of $\boldsymbol{h_i^k}$, where $\boldsymbol{h_i^k}$ is the hash code in the $k^{th}$ segment, $k \in \{1, ..., K\}$. Thus, the **weighted Hamming distance** between images $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ can be defined as:

$$D(\boldsymbol{h_i}, \boldsymbol{h_j}) = \frac{1}{2}(L - \sum_{k=1}^{K} u_k (\boldsymbol{h_i^k})^T \boldsymbol{h_j^k}). \tag{4}$$

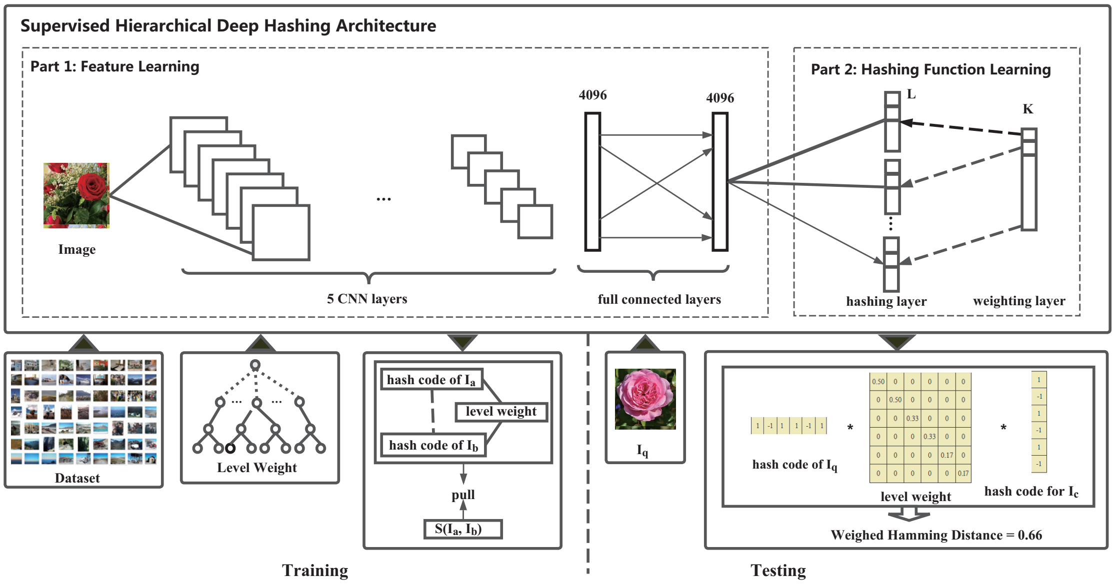


Figure 2: The SHDH learning framework. It takes raw images as input. The training stage is illustrated in the left bottom corner. A retrieval example in testing stage is presented in the right bottom corner.

We define the similarity-preserving objective function:

$$E_1 = \sum_{k=1}^{K} (\frac{1}{L_k} u_k (\boldsymbol{h_i^k})^T \boldsymbol{h_j^k} - s_{ij}^k)^2. \quad (5)$$

Eq. (5) is used to make sure the similar images could share same hash code in each segment, and it should be minimized.

Second, to maximize the information from each hash bit, each bit should be a balanced partition of the dataset (Do, Doan, and Cheung 2016). Thus, we use the entropy to measure the balanced ability, just as below:

$$E_2 = \sum_{k=1}^{K} u_k tr(\boldsymbol{h_i^k} (\boldsymbol{h_j^k})^T). \quad (6)$$

$E_2$ should be maximized which means " $-E_2$" should be minimized. Thus, we ultimately combine Eq. (5) and Eq. (6) to obtain the total objective function:

$$\begin{aligned} J &= \min E_1 - \alpha E_2 \\ &= \min \sum_{k=1}^{K} (\frac{1}{L_k} u_k (\boldsymbol{h_i^k})^T \boldsymbol{h_j^k} - s_{ij}^k)^2 \\ &\quad - \alpha \sum_{k=1}^{K} u_k tr(\boldsymbol{h_i^k} (\boldsymbol{h_j^k})^T), \end{aligned} \quad (7)$$

where $\alpha$ is hyper-parameter.

**Learning** Assume that $\boldsymbol{H}$ is all the hash code for $N$ data points where $\boldsymbol{H} = [\boldsymbol{h_1}, \boldsymbol{h_2}, ..., \boldsymbol{h_N}]^T$. Since the elements in $\boldsymbol{H}$ are discrete integer, $J$ is not derivable. So, we relax it as $\boldsymbol{B}$ from discrete to continuous by removing the sign function. Thus the objective function Eq. (7) could be transformed into the matrix form as below:

$$J = \min \|\boldsymbol{BAB}^T - L\boldsymbol{S}\|_F^2 - \alpha tr(\boldsymbol{BAB}^T), \quad (8)$$

where $\boldsymbol{A} \in \mathbb{R}^{L \times L}$ is a diagonal matrix. It can be divided into $K$ small diagonal matrix whose size is $L_k \times L_k$, and $\{A_{k_{jj}}\}_{j=1}^{L_k} = u_k$.

Stochastic gradient descent (SGD) is used to learn the parameters. The learning rate $\eta$ is initialized as 0.01, and updated by $\eta \leftarrow \frac{2}{3}\eta$ empirically. In particular, a minibatch data will be sampled in each iteration for learning. The derivative of Eq.(8) with respect to $\boldsymbol{B}$ is given by:

$$\begin{aligned} \frac{\partial J}{\partial \boldsymbol{B}} =& 2(\boldsymbol{BA}^T\boldsymbol{B}^T\boldsymbol{BA} + \boldsymbol{BAB}^T\boldsymbol{BA}^T \\ & - \boldsymbol{S}^T\boldsymbol{BA} - \boldsymbol{SBA}^T) - \alpha\boldsymbol{BA}^T - \alpha\boldsymbol{BA}. \end{aligned}$$

Then, the derivative value can be fed into the underlying network via the back-propagation (BP) algorithm to update all parameters.

The outline of the proposed supervised hierarchical deep hashing (SHDH) is described in Algorithm 1.

# Experiments

## Datasets and Setting

We carried out experiments on two public benchmark datasets: CIFAR-100 and IAPRTC-12. CIFAR-100 is an image dataset containing 60,000 colour images of 32×32 pixels. It has 100 classes and each class contains 600 images. The 100 classes in the CIFAR-100 are grouped into 20 superclasses. Each image has a "fine" label (the class which

**Algorithm 1** The Learning Algorithm for SHDH

**Require:** Training images $\boldsymbol{X} = \{\boldsymbol{x_i}\}_{i=1}^{N}$, the hierarchical labels of the training images, the length of hash code $L$, the number of levels in the hierarchy $K$, the max iterative count $T$, the size of minibatch (default 128).
**Ensure:** The hash code for all data points.

```
Initialize the weights and bias of whole network.
Initialize the learning rate η as 0.01.
S ← using Eq. (3), S ∈ ℝ^{N×N}.
repeat
    Update η ← 2η/3 every 20 iterations empirically.
    Randomly sample from X to get a minibatch.
    For each image x_i, perform as below:
    for k = 1, ··· , K do
        Calculate the output h_i^k of image x_i by forward propagation.
    end for
    Merge {h_i^k}_{k=1}^K, to get H.
    Update the parameters {W, v, θ} by back propagation.
until up to T
Put all data points into the network to calculate their hash code.
```

it belongs to) and a "coarse" label (the superclass which it belongs to). Thus, the height of the hierarchical labels with a "Root" node in CIFAR-100 is three. The IAPRTC-12 dataset has 20,000 segmented images. Each image has been manually segmented, and the resultant regions have been annotated according to a predefined vocabulary of labels. The vocabulary is organized according to a hierarchy of concepts. The height of the hierarchical labels in IAPRTC-12 is seven. For both datasets, we randomly selected 90% as the training set and the left 10% as the test set. The hyper-parameter $\alpha$ in SHDH is empirically set as one. The weights and bias in the feature learning part are initialized as the values pre-trained in VGG-F (Chatfield et al. 2014). The parameters including weights $\boldsymbol{W}$ and bias $\boldsymbol{v}$ in the hashing layer are initialized to be a quite small real number between 0 and 0.001 empirically. The learning rate $\eta$ is initialized as 0.01.

We compared our methods with six state-of-the-art hashing methods, where four of them are supervised, the other two are unsupervised. The four supervised methods include DPSH (Li, Wang, and Kang 2016), COSDISH (Kang, Li, and Zhou 2016), LFH (Zhang et al. 2014), and KSH (Liu et al. 2012). The two unsupervised methods are KMH (He, Wen, and Sun 2013) and ITQ (Gong and Lazebnik 2011). For all of these six baselines, we employed the implementations provided by the original authors, and used the default parameters recommended by the corresponding papers. Moreover, to study the influence of hierarchical labels separately, we replaced the values in the similarity matrix for KSH and COSDISH by using hierarchical similarity to obtain two new methods, KSH+H and COSDISH+H. "H" means hierarchical version. ITQ and KMH cannot be modified as "H"-version because they are unsupervised. LFH and DPSH cannot be modified as "H"-version because when similarity is a real value except zero and one, the likelihood functions used in their papers are meaningless.

We resized all images to 224×224 pixels and directly used the raw images as input for the deep hashing methods including SHDH and DPSH. The left six methods use hand-crafted features as stated on the original papers. We represented each image by a 512-D GIST vector.

## Evaluation Criterion

Cai (Cai 2016) has claimed that the performance of some hashing algorithms (e.g., LSH) can easily be boosted if one uses multiple hash tables, which is an important factor should be considered in the evaluation while most of the existing papers failed to correctly measure the search time which is essential for the ANN search problem. However, hash table is not a standard configuration for hashing, which can be replaced by other data structures. In addition, data structures, such as the number of hash tables, the index of hash tables and the hierarchy of hash tables (Liu, He, and Lang 2013)), will have a significant impact on the search time evaluation. The main purpose of our work is to study the impact of the hierarchy in hierarchical labeled data on hashing, not focus on the efficiency which can be studied by other researches. Thus, we focus on the effectiveness evaluation, and in order to ensure the fairness of evaluations, the distance between a data point and a query will be calculated by brute-force search for all methods and different code length.

To verify the effectiveness of hash code, we measured the ranking quality of retrieved list for different methods by Average Cumulative Gain (ACG), Discounted Cumulative Gain (DCG), Normalized Discounted Cumulative Gain (NDCG) (Järvelin and Kekäläinen 2000), and Weighted Recall. Note that we proposed the Weighted Recall metric to measure the recall in the scenario of hierarchical labeled data, defined as:

$$Weighted\ Recall(q)@n = \frac{\sum_{i=1}^{n} s_{qi}}{\sum_{i=1}^{N} s_{qi}},$$

where $n$ is the number of top returned data points, $s_{qi}$ represents the similarity between the query $q$ and $i^{th}$ data point in the ranking list, $N$ is the length of the ranking list.

## Results on CIFAR-100

Table 1 summarizes the comparative results of different hashing methods on the CIFAR-100 dataset. We have several observations from Table 1: (1) our SHDH outperforms the other supervised and unsupervised baselines for different code length. For example, comparing with the best competitor (DPSH), the results of our SHDH have a relative increase of 12.5% $\sim$ 18.4% on ACG, 10.7% $\sim$ 16.7% on DCG, and 8.7% $\sim$ 11.4% on NDCG; (2) the hierarchical semantic labels can improve the performance of hashing methods. For example, COSDISH+H and KSH+H perform respectively better than COSDISH and KSH, which means the inherent hierarchical information is valuable to improve hashing performance; (3) among all the supervised approaches, the

Table 1: Results on the CIFAR-100 dataset. The ranking results are measured by ACG, DCG, and NDCG@N (N=100).

| Methods | ACG@100 | | | DCG@100 | | | NDCG@100 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 32 | 48 | 64 | 32 | 48 | 64 | 32 | 48 | 64 |
| KMH | 0.2023 | - | 0.2261 | 6.0749 | - | 6.7295 | 0.4169 | - | 0.4189 |
| ITQ | 0.2091 | 0.2312 | 0.2427 | 6.1814 | 6.7583 | 7.0593 | 0.4197 | 0.4243 | 0.4272 |
| COSDISH+H | 0.1345 | 0.1860 | 0.2008 | 4.2678 | 5.5619 | 5.9169 | 0.4072 | 0.4417 | 0.4523 |
| KSH+H | 0.1611 | 0.1576 | 0.1718 | 4.9904 | 4.9282 | 5.3378 | 0.3940 | 0.3897 | 0.3924 |
| DPSH | 0.4643 | 0.4973 | 0.5140 | 11.5129 | 12.2878 | 12.7072 | 0.5650 | 0.5693 | 0.5751 |
| COSDISH | 0.1366 | 0.1428 | 0.1501 | 4.5079 | 4.6957 | 4.8601 | 0.4063 | 0.4156 | 0.4127 |
| LFH | 0.1152 | 0.1291 | 0.1271 | 3.7847 | 4.3299 | 4.3239 | 0.3924 | 0.4008 | 0.4011 |
| KSH | 0.1291 | 0.1393 | 0.1509 | 3.3520 | 4.3009 | 4.8293 | 0.3711 | 0.3766 | 0.3763 |
| SHDH | **0.5225** | **0.5724** | **0.6084** | **12.7460** | **13.9575** | **14.7861** | **0.6141** | **0.6281** | **0.6406** |

Table 2: Results on the IAPRTC-12 dataset. The ranking results are evaluated by ACG, DCG, and NDCG@N (N=100).

| Methods | ACG@100 | | | DCG@100 | | | NDCG@100 | | |
|---|---|---|---|---|---|---|---|---|---|
| | 48 | 64 | 128 | 48 | 64 | 128 | 48 | 64 | 128 |
| KMH | - | 3.7716 | 3.7446 | - | 87.5121 | 87.0493 | - | 0.6427 | 0.6373 |
| ITQ | 3.8351 | 3.8502 | 3.8609 | 88.5562 | 88.9057 | 89.2016 | 0.6626 | 0.6633 | 0.6652 |
| COSDISH+H | 3.8249 | 3.7245 | 3.8448 | 88.3121 | 86.3037 | 88.5056 | 0.6957 | 0.6885 | 0.6970 |
| KSH+H | 3.7304 | 3.7535 | 3.7779 | 86.5606 | 87.0894 | 87.5743 | 0.6459 | 0.6494 | 0.6518 |
| DPSH | 4.0085 | 4.0227 | 4.0980 | 91.4972 | 92.0570 | 93.4613 | 0.6618 | 0.6607 | 0.6630 |
| COSDISH | 3.6856 | 3.6781 | 3.7018 | 85.2368 | 85.1622 | 85.7606 | 0.6412 | 0.6443 | 0.6408 |
| LFH | 3.7076 | 3.6851 | 3.6988 | 85.7599 | 85.2662 | 85.6601 | 0.6390 | 0.6365 | 0.6400 |
| KSH | 3.8357 | 3.8317 | 3.7909 | 88.5041 | 88.5589 | 87.8282 | 0.6507 | 0.6482 | 0.6408 |
| SHDH | **4.4870** | **4.5284** | **4.5869** | **100.6373** | **101.4812** | **102.6919** | **0.7372** | **0.7440** | **0.7489** |

deep learning based approaches (SHDH and DPSH) give relatively better results, and it confirms that the learnt representations by deep network from raw images are more effective than hand-crafted features to learn hash code.

Figure 3 (a) $\sim$ (c) are the Weighted Recall curves for different methods over different weighted Hamming distance at 32, 48, and 64 bits, respectively, which shows our method has a consistent advantage over baselines. Figure 3 (g) $\sim$ (i) are the Weighted Recall results over top-$n$ retrieved results, where $n$ ranges from 1 to 5,000. Our approach also outperforms other state-of-the-art hashing methods. The Weighted Recall curves at different length of hash code are also illustrated in Figure 4 (a). From the figure, our SHDH model performs better than baselines, especially when the code length increases. This is because when the code length increases, the learnt hash functions can increase the discriminative ability for hierarchical similarity among images.

### Results on IAPRTC-12

Table 2 shows the performance comparison of different hashing methods over IAPRTC-12 dataset, and our SHDH performs better than other approaches regardless of the length of codes. Obviously, it can be found that all baselines cannot achieve optimal performance for hierarchical labeled data. Figure 3 (j) $\sim$ (l) are the Weighted Recall results over top-$n$ returned neighbors, where $n$ ranges from 1 to 5,000. These curves show a consistent advantage against baselines. Moreover, our SHDH provides the best performance at different code length, shown in Figure 4 (b).

The results of the Weighted Recall over different weighted Hamming distance are shown in Figure 3 (d) $\sim$ (f). In these figures, our method is not the best one. The reason is that our SHDH has better discriminative ability at the same weighted Hamming distance due to considering the hierarchical relation. For example, DPSH returns 4,483 data points while our SHDH only returns 2,065 points when the weighted Hamming distance is zero and the code length is 64 bits. Thus, the better discriminative ability leads to better precision (Table 2) but not-so-good Weighted Recall.

### Sensitivity to Hyper-Parameter

Figure 5 shows the effect of the hyper-parameter $\alpha$ over CIFAR-100. We can find that SHDH is not sensitive to $\alpha$. For example, SHDH can achieve good performance on both datasets with $0.5 \leq \alpha \leq 2$. We can also obtain similar conclusion over IAPRTC-12 dataset, and the figure is not included in this paper due to the limitation of space.

## Conclusion

In this paper, we have proposed a novel supervised hierarchical deep hashing method for hierarchical labeled data. To the best of our knowledge, SHDH is the first method to utilize the hierarchical labels of images in supervised hashing area. Extensive experiments on two real-world public datasets have shown that the proposed SHDH method outperforms the state-of-the-art hashing algorithms.

In the future, we will further verify the effectiveness of the proposed SHDH method on larger datasets, such as ImageNet. Moreover, we will also explore more deep hashing

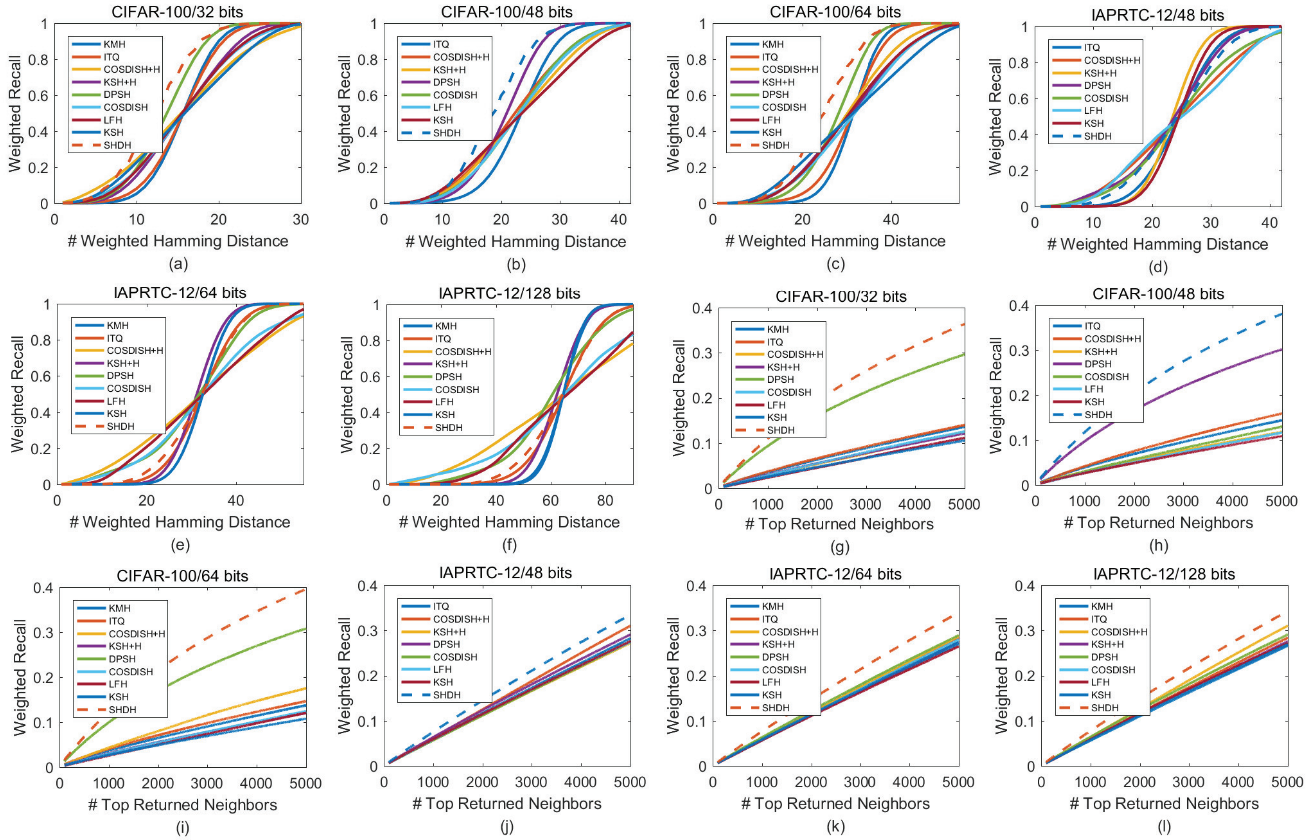


Figure 3: Weighted Recall curves on CIFAR-100 and IAPRTC-12. (a) $\sim$ (f) show the Weighted Recall within various weighted Hamming distance at different number of bits. (g) $\sim$ (l) show the Weighted Recall@n at different number of bits.

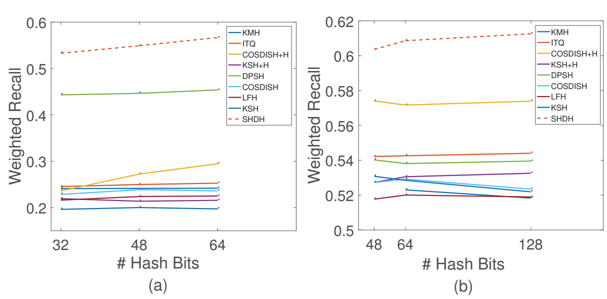


Figure 4: Weighted Recall@n (n=10,000) over (a) CIFAR-100 and (b) IAPRTC-12.

methods to process hierarchical labeled data. What's more, the performance of hashing methods for non-hierarchical labeled data will be further improved by constructing their hierarchy automatically.

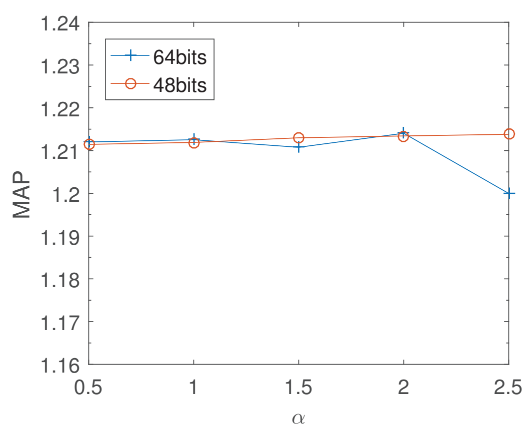


Figure 5: Sensitivity to hyper-parameter $\alpha$ over CIFAR-100.

## Acknowledgments


This work was supported by 863 Program (2015AA015404), China National Science Foundation (61402036, 61772076, 60973083, 61273363), Science and Technology Planning Project of Guangdong Province (2014A010103009, 2015A020217002), Guangzhou Science and Technology Planning Project (201604020179), Beijing Advanced Innovation Center for Imaging Technology (BAICIT-2016007) and Open Fund Project from Beijing Key Laboratory of Internet Culture and Digital Dissemination Research (ICDD201701).

## References


Andoni, A., and Indyk, P. 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM* 51(1):117–122.

Cai, D. 2016. A revisit of hashing algorithms for approximate nearest neighbor search. *CoRR* abs/1612.07545.

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

Datar, M.; Immorlica, N.; Indyk, P.; and Mirrokni, V. S. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the annual Symposium on Computational geometry*, 253–262.

Deng, J.; Dong, W.; Socher, R.; Li, L.; Li, K.; and Li, F. 2009. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 248–255. Miami, Florida, USA: IEEE Computer Society.

Do, T.; Doan, A.; and Cheung, N. 2016. Learning to hash with binary deep neural network. In *European Conference on Computer Vision*, 219–234. Amsterdam, The Netherlands: Springer.

Escalante, H. J.; Hernández, C. A.; González, J. A.; López-López, A.; Montes-y-Gómez, M.; Morales, E. F.; Sucar, L. E.; Pineda, L. V.; and Grubinger, M. 2010. The segmented and annotated IAPR TC-12 benchmark. *Computer Vision and Image Understanding* 114(4):419–428.

Gionis, A.; Indyk, P.; and Motwani, R. 1999. Similarity search in high dimensions via hashing. In *International Conference on Very Large Data Bases*, 518–529. Edinburgh, Scotland, UK: Morgan Kaufmann.

Gong, Y., and Lazebnik, S. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *Conference on Computer Vision and Pattern Recognition*, 817–824. Colorado Springs, CO, USA: IEEE Computer Society.

He, K.; Wen, F.; and Sun, J. 2013. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Conference on Computer Vision and Pattern Recognition*, 2938–2945. Portland, OR, USA: IEEE Computer Society.

Järvelin, K., and Kekäläinen, J. 2000. IR evaluation methods for retrieving highly relevant documents. In *Conference on Research and Development in Information Retrieval*, 41–48. Athens, Greece: ACM.

Kang, W.; Li, W.; and Zhou, Z. 2016. Column sampling based discrete supervised hashing. In *Conference on Artificial Intelligence*, 1230–1236. Phoenix, Arizona, USA: AAAI Press.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Conference on Neural Information Processing Systems*, 1106–1114.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. In *University of Toronto*.

Kulis, B.; Jain, P.; and Grauman, K. 2009. Fast similarity search for learned metrics. *IEEE Trans. Pattern Anal. Mach. Intell.* 31(12):2143–2157.

Li, X.; Lin, G.; Shen, C.; van den Hengel, A.; and Dick, A. R. 2013. Learning hash functions using column generation. In *International Conference on Machine Learning*, 142–150. Atlanta, GA, USA: JMLR.org.

Li, W.; Wang, S.; and Kang, W. 2016. Feature learning based deep supervised hashing with pairwise labels. In *International Joint Conference on Artificial Intelligence*, 1711–1717. New York, NY, USA: IJCAI/AAAI Press.

Liu, W.; Wang, J.; Ji, R.; Jiang, Y.; and Chang, S. 2012. Supervised hashing with kernels. In *Conference on Computer Vision and Pattern Recognition*, 2074–2081.

Liu, X.; He, J.; and Lang, B. 2013. Reciprocal hash tables for nearest neighbor search. In *Conference on Artificial Intelligence Bellevue*. Washington, USA: AAAI Press.

Qiu, Z.; Pan, Y.; Yao, T.; and Mei, T. 2017. Deep semantic hashing with generative adversarial networks. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, 225–234.

Raginsky, M., and Lazebnik, S. 2009. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems*, 1509–1517.

Shen, F.; Liu, W.; Zhang, S.; Yang, Y.; and Shen, H. T. 2015. Learning binary codes for maximum inner product search. In *International Conference on Computer Vision*, 4148–4156. Santiago, Chile: IEEE Computer Society.

Wang, J.; Kumar, O.; and Chang, S. 2010. Semi-supervised hashing for scalable image retrieval. In *Conference on Computer Vision and Pattern Recognition*, 3424–3431. San Francisco, CA, USA: IEEE Computer Society.

Yuan, X.; Lu, J.; Chen, Z.; Feng, J.; and Zhou, J. 2017. Reconstruction-based supervised hashing. In *IEEE International Conference on Multimedia and Expo*, 1404–1409.

Zhang, C., and Zheng, W. 2017. Semi-supervised multi-view discrete hashing for fast image search. *IEEE Trans. Image Processing* 26(6):2604–2617.

Zhang, P.; Zhang, W.; Li, W.; and Guo, M. 2014. Supervised hashing with latent factor models. In *International Conference on Research and Development in Information Retrieval*, 173–182. Gold Coast, QLD, Australia: ACM.

Zhang, J.; Peng, Y.; and Zhang, J. 2016. SSDH: semi-supervised deep hashing for large scale image retrieval. *CoRR* abs/1607.08477.

Zheng, L.; Wang, S.; Tian, L.; He, F.; Liu, Z.; and Tian, Q. 2015. Query-adaptive late fusion for image search and person re-identification. In *Conference on Computer Vision and Pattern Recognition*, 1741–1750.

Zhu, H.; Long, M.; Wang, J.; and Cao, Y. 2016. Deep hashing network for efficient similarity retrieval. In *Conference on Artificial Intelligence*, 2415–2421. Phoenix, Arizona, USA: AAAI Press.

Zhu, L.; Shen, J.; Xie, L.; and Cheng, Z. 2017. Unsupervised visual hashing with semantic assistant for content-based image retrieval. *IEEE Trans. Knowl. Data Eng.* 29(2):472–486.