Object Detection based Deep Unsupervised Hashing

Rong-Cheng Tu[1,2] , **Xian-Ling Mao**[*1,3] , **Bo-Si Feng**[1] , **Shu-ying Yu**[1] ,
[1]Department of Computer Science and Technology, Beijing Institute of Technology, China
[2]CETC Big Data Research Institute, China
[3]Zhijiang Lab, China
{tu_rc,maoxl,2120160986,syyu}@bit.edu.cn

Abstract

Recently, similarity-preserving hashing methods have been extensively studied for large-scale image retrieval. Compared with unsupervised hashing, supervised hashing methods for labeled data have usually better performance by utilizing semantic label information. Intuitively, for unlabeled data, it will improve the performance of unsupervised hashing methods if we can first mine some supervised semantic 'label information' from unlabeled data and then incorporate the 'label information' into the training process. Thus, in this paper, we propose a novel Object Detection based Deep Unsupervised Hashing method (ODDUH). Specifically, a pre-trained object detection model is utilized to mining supervised 'label information', which is used to guide the learning process to generate high-quality hash codes. Extensive experiments on two public datasets demonstrate that the proposed method outperforms the state-of-the-art unsupervised hashing methods in the image retrieval task.

1 Introduction

With the rapid growth of image data, approximate nearest neighbour (ANN) search have attracted more and more attention from researchers in the large-scale image search area. Among the existing ANN search techniques, similarity-preserving hashing methods are advantageous due to their high retrieval efficiency and low storage cost. The main idea of hashing methods is to transform high dimensional data points into a set of compact binary codes, meanwhile, maintain similarity of the original data points. Since the original data points are represented by binary codes instead of real valued features, the searching time and storage cost can be dramatically reduced.

In general, data-dependent hashing can be divided into unsupervised [Gong and Lazebnik, 2011; Huang *et al.*, 2017; Ghasedi Dizaji *et al.*, 2018; Lin *et al.*, 2016] and supervised [Wang *et al.*, 2018; Qiu *et al.*, 2017] methods. The unsupervised hashing methods mainly utilize the features of images to generate similarity-preserving binary codes without any supervised information. Compared with unsupervised hashing, supervised hashing methods incorporate semantic labels of training data into training process, thus they can perform more remarkably in generating similarity-preserving binary code. However, in many real applications, there are no semantic labels for images, and supervised hashing methods cannot be used in these cases

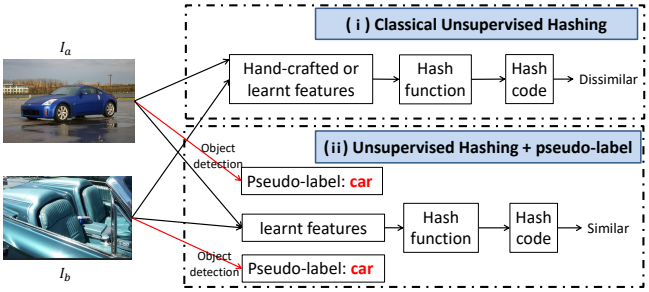


Figure 1: High-quality similarity-preserving hashing code can be produced by utilizing the pseudo-labels mined from images. The block (i) is the workflow of the existing unsupervised hashing methods which do not mine the pseudo-labels from images, it is hard for them to judge that the two images I_a and I_b are similar. However, in block (ii), we use pseudo-labels mined from images to train hashing models which can easily judge the two image are similar.

Intuitively, if we can detect the objects in images and use their corresponding classes as the pseudo-labels of the images, then we can use the pseudo-labels as 'supervised information' to guide hash codes learning to obtain better performance. An illustrative example is shown in Figure 1, the block (i) is the procedure of existing unsupervised hashing methods. They use the hand-crafted or learnt features as inputs, and then they directly use the Euclidean distance between images or the similarity between one image and its rotated image to guide the hash training, which will make the existing unsupervised hashing models to judge that the images I_a and I_b are dissimilar with high possibility. Actually, the images I_a and I_b are similar, because the two images belong to the class 'car'. On the contrary, in the block (ii), if we first use an object detection model to get the pseudo-labels 'car'of the two images by detecting objects inside an image. By utilizing the pseudo-labels, we can construct pairwise similarity to guide hashing learning and make the hash-

*The author is corresponding author

ing models to judge that the two images I_a and I_b are similar with high possibility.

Inspired by this idea, we propose a novel Object Detection based Deep Unsupervised Hashing model, called ODDUH. In particular, an object detection model is first pre-trained on a large database. Then, we utilize the object detection model to mine latent semantic 'label information' (i.e., pseudo-labels) from images. Note that some of the pseudo-labels obtained by object detection are wrong because the precision of object detection is not 100%, and the noisy label will damage the final hashing performance. In order to decrease the harm of noisy pseudo-labels, we define a novel similarity criterion called pair-wise percentage similarity. Furthermore, a shared CNN is introduced to capture the feature representations of images. Finally, the pair-wise percentage similarity and the learnt feature representations of images are used to learn hash functions which can generate high-quality similarity-preserving hash codes. Extensive experiments on two real-world public datasets illustrate that our method outperforms the state-of-the-art unsupervised hash methods in image retrieval task.

2 Related Work

Generally, existing hashing methods can be divided into two categories: data-independent hashing and data-dependent hashing. For data-independent hashing methods, the hashing functions depend on random projection without any training data. The representative data-independent methods include Locality Sensitivity Hashing (LSH) [Gionis *et al.*, 1999] and its variants [Datar *et al.*, 2004; Kulis *et al.*, 2009]. For data-dependent hashing methods, they can achieve better accuracy with shorter codes by learning hash functions from training data. Furthermore, data-dependent methods can be mainly classified into two categories: supervised [Wang *et al.*, 2018; Qiu *et al.*, 2017] and unsupervised [Gionis *et al.*, 1999; Jin *et al.*, 2014; Ghasedi Dizaji *et al.*, 2018] methods.

The supervised data-dependent methods can achieve remarkable performance by utilizing labeled data to learn hashing functions. The representative methods in this category include Deep Supervised Hashing with Pairwise (DPSH) [Li *et al.*, 2015], HashNet [Cao *et al.*, 2017] and Deep Semantic Hashing with GANs (DSH-GANs) [Qiu *et al.*, 2017], etc. DPSH utilizes pair-wise similarity to guide hash functions learning in an end-to-end deep network. HashNet alleviates data imbalance that the number of similar pairs is much smaller than the number of dissimilar pairs by adjusting the weights of semantic similarity matrix to learn discriminative hashing codes. DSH-GANs introduces generative adversarial network (GAN) to generative fake images and construct triplet with the fake images and real images to supervise hash functions learning process.

The unsupervised data-dependent methods can be divided as traditional unsupervised hashing methods and deep unsupervised hashing methods. The traditional unsupervised hashing methods use hand-crafted features and shallow hash functions to obtain binary hash code. Lots of algorithms in this category have been proposed, including Spectral Hashing (SH) [Weiss *et al.*, 2009], and Iterative Quantization (ITQ) [Gong and Lazebnik, 2011]. However, limited by the hand-crafted features and shallow hash functions, it is hard for them to deal with complex and high dimensional real-world data and keep the semantic similarity between original data in the binary hash codes. The deep unsupervised hashing methods utilize deep architecture to extract image features to learn hash code. For example, Deepbit [Lin *et al.*, 2016] get rotation invariant and balanced binary hash codes by defined a quantization loss. Unsupervised triplet hashing (UTH) [Huang *et al.*, 2017] employs an unsupervised triplet loss to get balanced hash codes. HashGAN [Ghasedi Dizaji *et al.*, 2018] generate compact hash codes by a generative adversarial hashing network.

However, few existing unsupervised hashing methods try to mine the latent semantic 'label information' in the images. Thus, in this paper, we propose a novel deep unsupervised hashing model based on object detection to generate high-quality hash codes by incorporating the latent semantic 'label information' mined by object detection into the training process.

3 Object Detection based Deep Unsupervised Hashing Method

In this section, we will present the proposed Object Detection based Deep Unsupervised Hashing Network (ODDUH) in detail. We first give a description of model notation in section 3.1. The whole architecture of ODDUH will be introduced in section 3.2. Then we discuss the detail of similarity definition and the object function in section 3.3 and section 3.4, respectively. Finally, we will introduce the learning of parameters in section 3.5.

3.1 Notation

Suppose a dataset has n images $X = \{x_i\}_{i=1}^n$, and the i^{th} image is x_i. The goal of similarity-preserving hashing is to learn a mapping $H : x_i \rightarrow \mathbf{b}_i \in \{-1, 1\}^k$, where k is the length of hashing codes, such that an input image x_i will be encoded into a k-bit binary code $\mathbf{b}_i$.

3.2 The Architecture of ODDUH

As shown in Figure 2, our architecture consists of three parts: latent semantic 'label information' mining, feature learning and hash function learning.

In the latent semantic 'label information' mining part, the ODDUH uses a pre-trained object detection model named YOLOv2 [Redmon and Farhadi, 2017] to mining the latent semantic 'label information' in images. Note that other state-of-the-art object detection models can also be used here, such as SSD [Liu *et al.*, 2016] and Mask R-CNN [He *et al.*, 2017].

The feature learning part includes a convolutional neural network (CNN) component and two fully-connected layers. The CNN component contains five convolutional layers. Particularly, the first convolutional layer filters the input images with 96 kernels of size 11×11 with a stride of four pixels. The output of the first convolutional layer will be response-normalized and maxpooled (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

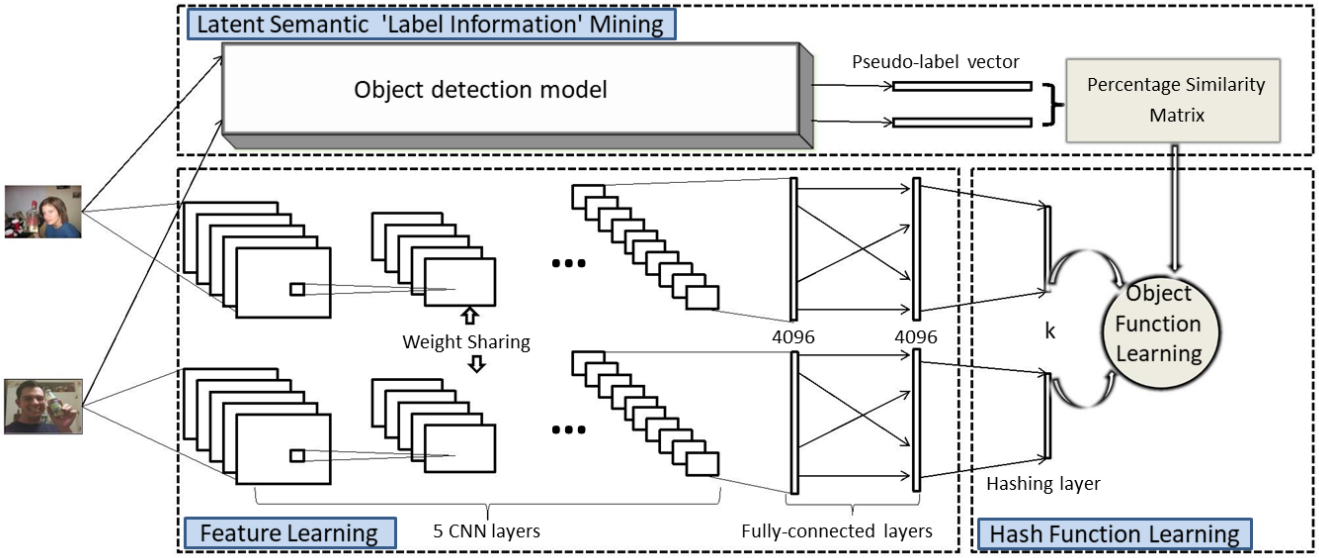


Figure 2: The ODDUH learning framework. The Mining Latent Semantic 'Label Information' is a pre-trained object detection model. It is used to get the pseudo-labels. A shared CNN is implemented for learning image feature representations in the Feature Learning part. In Hash Function Learning part, a pair-wise loss function with Percentage Similarity Matrix is minimized to get the optimal hash function.

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 384 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 *et al.*, 2012].

The hash function learning part is a hashing layer which has k units. Furthermore, k is the length of hash code, and the hashing functions are learnt by the hashing layer. Eventually, we use element-wise sign function $sgn(\cdot)$, which returns 1 if the element is positive and returns -1 otherwise, to process the outputs of the hashing layer and get the binary code $\mathbf{b}$.

3.3 Pair-wise Percentage Similarity

In ODDUH, we use an object detection model to mine the objects in images and get their classes (i.e., pseudo-labels). Note that some of the pseudo-labels obtained by object detection are wrong because the precision of object detection is not 100%. If two images share at least one pseudo-label, then we let the similarity of them equal to 1, and equal to 0 otherwise(i.e., two-value similarity). It will easily make two dissimilar images as a similar pair which will damage the final hashing performance. Thus, in order to decrease the harm of noisy pseudo-labels, a novel pair-wise percentage similarity is defined as:

$$s_{ij} = \frac{\langle l_i, l_j \rangle}{\|l_i\|_1 + \|l_j\|_1 - \langle l_i, l_j \rangle} \tag{1}$$

where $\langle l_i, l_j \rangle$ calculate the inner product and $l_i \in \{0,1\}^c$ is the pseudo-label vector of x_i, where c is the total number of classes that pseudo-labels belong to. If i^{th} image x_i has the j^{th} pseudo-label, then $l_{ij} = 1$, else $l_{ij} = 0$. And $\|\cdot\|_1$ is the L_1-norm.

By incorporating the pair-wise percentage similarity into the training process, the learnt binary codes $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^n$ can preserve the similarity in $S = \{s_{ij} | i, j \in \{1, 2, \ldots, n\}, s_{ij} \in [0, 1]\}$. More specifically, if $s_{ij} = 0$, the binary codes $\mathbf{b}_i$ and $\mathbf{b}_j$ should have large Hamming distance; If $s_{ij} = 1$, the binary codes $\mathbf{b}_i$ and $\mathbf{b}_j$ should have a small Hamming distance; Otherwise, the binary codes $\mathbf{b}_i$ and $\mathbf{b}_j$ should have a suitable Hamming distance complying with the similarity s_{ij}.

3.4 Objective Function

Given the binary codes $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^n$ for all the images, we can define the likelihood of the pair-wise percentage similarity s_{ij} as:

$$p(s_{ij}|\mathbf{B}) = \begin{cases} \sigma(t_{ij}\Psi_{ij}), & s_{ij} = 1 \; or \; 0, \\ 1 - |s_{ij} - \sigma(\Psi_{ij})|, & otherwise. \end{cases} \tag{2}$$

where $t_{ij} = 2s_{ij} - 1$, $\Psi_{ij} = \frac{1}{2}\mathbf{b}_i^T\mathbf{b}_j$, and $\sigma(x) = \frac{1}{1+e^{-x}}$. When $s_{ij} = 0 \; or \; 1$, we take the negative log-likelihood of the observed pair-wise labels in S to measure the pair-wise similarity loss; When $0 < s_{ij} < 1$, we take mean square error to measure the pair-wise similarity loss. Thus, the pair-wise similarity loss function can be defined as:

$$\begin{aligned} L_1 = \sum_{s_{ij} \in S} [&-\alpha \cdot I_{ij} log(\sigma(t_{ij}\Psi_{ij})) \\ &+ (1 - I_{ij})(s_{ij} - \sigma(\Psi_{ij}))^2] \end{aligned} \tag{3}$$

where α is a hyper-parameter. I_{ij} is used to denote two conditions, which is defined as:

$$I_{ij} = \begin{cases} 1, & s_{ij} = 1 \; or \; 0, \\ 0, & otherwise. \end{cases} \tag{4}$$

By minimizing Eq. (3), we can make the hamming distance between two completely similar points as small as possible, and simultaneously make the hamming distance between two dissimilar points as large as possible. Meanwhile, we can make the partly similar images x_i and x_j have the suitable hamming distance complying with the similarity s_{ij}.

However, Eq. (3) is a discrete optimization problem, which is difficult to solve. Following the idea of previous work [Li *et al.*, 2015], we relax $\mathbf{b}$ from discrete to continuous, and then reformulate Eq. (3) as:

$$L_2 = \sum_{s_{ij} \in S} [\alpha \cdot I_{ij} log(1 + e^{-t_{ij}\Theta_{ij}}) + (1 - I_{ij})(s_{ij} - \sigma(\Theta_{ij}))^2] \tag{5}$$

where $\Theta_{ij} = \frac{1}{2}\mathbf{u}_i^T \mathbf{u}_j$. $\mathbf{u}_i \in \mathbb{R}^k$ is the outputs of hashing layer: $\mathbf{u}_i = \mathbf{W}^T \mathcal{F}(\mathbf{x}_i; \boldsymbol{\theta}) + \mathbf{v}$, where the mapping $\mathcal{F} : \mathbb{R}^d \rightarrow \mathbb{R}^{4096}$ is parameterized by $\boldsymbol{\theta}$, and $\boldsymbol{\theta}$ represents the parameters of the seven layers of network in the feature learning part. $\mathbf{W} \in \mathbb{R}^{4096 \times k}$ is the weight matrix to be learnt at the hashing layer, $\mathbf{v} \in \mathbb{R}^k$ is the bias. Due to the $\mathbf{u}_i$ is not the binary codes, we used a quantization loss to make $\mathbf{u}_i$ to be close to binary codes. The quantization loss is defined as:

$$L_q = \sum_{i}^{n} ||\mathbf{b}_i - \mathbf{u}_i||_2^2 \tag{6}$$

Then, by connecting the pseudo-label pair-wise similarity loss and quantization loss, the final objective function can be defined as:

$$L = L_2 + \beta L_q \tag{7}$$

where β is a hyper-parameter.

3.5 Learning

In our method, there are four types of parameters $\mathbf{B}, \mathbf{W}, \boldsymbol{\theta}, \mathbf{v}$ need to be learnt, during the training phase. A mini-batch gradient descent method is used as optimization algorithm. Moreover, we use an alternating method to learn the parameters. More specifically, we optimize one parameter with other parameters fixed.

With parameters $\mathbf{W}, \boldsymbol{\theta}, \mathbf{v}$ fixed, the $\mathbf{b}_i$ can be directly optimized as follows:

$$\mathbf{b}_i = sgn(\mathbf{u}_i) = sgn(\mathbf{W}^T \mathcal{F}(\mathbf{x}_i; \boldsymbol{\theta}) + \mathbf{v}) \tag{8}$$

With $\mathbf{b}_i$ fixed, the other parameters $\mathbf{W}, \boldsymbol{\theta}, \mathbf{v}$ are learned by standard back-propagation algorithm. Especially, we are able to compute the derivatives of the loss function about $\mathbf{u}_i$ as follows:

$$\begin{aligned}\frac{\partial L}{\partial \mathbf{u}_i} = & \sum_{j:s_{ij} \in S} [-\frac{1}{2}\alpha \cdot t_{ij} I_{ij}(1 - \sigma(t_{ij}\Theta_{ij})) \\ & + (1 - I_{ij})\sigma(\Theta_{ij})(1 - \sigma(\Theta_{ij}))(s_{ij} - \sigma(\Theta_{ij}))]\mathbf{u}_j \\ & + \sum_{j:s_{ji} \in S} [-\frac{1}{2}\alpha \cdot t_{ji} I_{ji}(1 - \sigma(t_{ji}\Theta_{ji})) \\ & + (1 - I_{ji})\sigma(\Theta_{ji})(1 - \sigma(\Theta_{ji}))(s_{ji} - \sigma(\Theta_{ji}))]\mathbf{u}_j \\ & + 2\beta(\mathbf{u}_i - \mathbf{b}_i)\end{aligned} \tag{9}$$

Then, we can use the standard back-propagation algorithm to update $\mathbf{W}, \boldsymbol{\theta}$ and $\mathbf{v}$ with Eq. (9):

$$\frac{\partial L}{\partial \mathbf{W}} = \mathcal{F}(\mathbf{x}_i; \boldsymbol{\theta})(\frac{\partial L}{\partial \mathbf{u}_i})^T \tag{10}$$

$$\frac{\partial L}{\partial \mathcal{F}(\mathbf{x}_i; \boldsymbol{\theta})} = \mathbf{W}\frac{\partial L}{\partial \mathbf{u}_i} \tag{11}$$

$$\frac{\partial L}{\partial \mathbf{v}} = \frac{\partial L}{\partial \mathbf{u}_i} \tag{12}$$

4 Experiments

4.1 Datasets and Baselines

We conduct experiments on two public benchmark datasets: Pascal VOC 2007 [1] [Everingham *et al.*, 2010] and BMVC 2009 [2] [Allan and Verbeek, 2009]. Pascal VOC 2007 consists of 9,963 multi-label images. There are 20 object classes in this dataset. On average, each image is annotated with 1.5 labels. BMVC 2009 contains 96,378 images collected from Flickr. Each image in the dataset is associated with one or multiple labels in 20 semantic concepts.

Our proposed method is an unsupervised method, thus we compare our method with eight classical and state-of-the-art unsupervised hashing methods including: LSH [Gionis *et al.*, 1999], ITQ [Gong and Lazebnik, 2011], SH [Weiss *et al.*, 2009], PCAH [Wang *et al.*, 2010], SGH [Jiang and Li, 2015], UH_BDNN [Do *et al.*, 2016], UTH [Huang *et al.*, 2017], and HashGAN [Ghasedi Dizaji *et al.*, 2018], where LSH, SH, ITQ, PCAH and SGH are traditional unsupervised methods and the other three are deep unsupervised methods. Note that the five traditional unsupervised hashing methods use hand-crafted features as inputs, i.e., each image in Pascal VOC 2007 and BMVC 2009 is represented by a 512-dimensional GIST vector. For the deep unsupervised hashing method UH_BDNN, it use the outputs of the fc7 layer in AlexNet as image representation. For the other two deep unsupervised hashing methods and our proposed method, we resize all the images to be 224×224 pixels and then directly use the raw image pixels as input. When carrying out experiments on the two datasets respectively, we randomly select 2,000 images as test set and the left images as training dataset. Moreover, in order to proves that the learnt representations by deep network from raw images are more superior than hand-crafted features in hash learning procedure, we also conduct the experiments by using the outputs of the fc7 layer in AlexNet [Krizhevsky *et al.*, 2012] as image representation in the five hand-crafted feature based hashing approaches and denote them as LSH+CNN, SH+CNN, ITQ+CNN and PCAH+CNN, respectively.

4.2 Implementation Details

For the object detection component, we choose YOLOv2 [Redmon and Farhadi, 2017]. It is pre-trained in COCO 2014 dataset which contains 81 object classes. Please note that all the object classes contained in Pascal VOC 2007 and BMVC

[1] http://host.robots.ox.ac.uk/pascal/VOC/voc2007/

[2] http://pascal.inrialpes.fr/data2/flickr-bmvc2009/

Methods	MAP@1000				WMAP@1000				NDCG@1000				ACG@1000			
	12bits	24bits	36bits	48bits	12bits	24bits	36bits	48bits	12bits	24bits	36bits	48bits	12bits	24bits	36bits	48bits
LSH	0.2676	0.2875	0.2916	0.2877	0.2881	0.3115	0.3168	0.3123	0.2230	0.2379	0.2436	0.2407	0.2821	0.2998	0.3009	0.2968
SH	0.3071	0.3021	0.3028	0.3023	0.3337	0.3287	0.3299	0.3299	0.2568	0.2514	0.2527	0.2530	0.3131	0.3074	0.3071	0.3055
PCAH	0.2884	0.2802	0.2783	0.2778	0.3124	0.3039	0.3018	0.3013	0.2384	0.2320	0.2307	0.2305	0.2982	0.2883	0.2849	0.2837
ITQ	0.2879	0.3086	0.3137	0.3223	0.3110	0.3345	0.3404	0.3509	0.2366	0.2584	0.2620	0.2718	0.2924	0.3191	0.3258	0.3358
SGH	0.3028	0.3081	0.3073	0.3107	0.3288	0.3358	0.3350	0.3395	0.2559	0.2611	0.2614	0.2644	0.3052	0.3082	0.3065	0.3089
LSH+CNN	0.2924	0.3351	0.3611	0.3694	0.3226	0.3737	0.4001	0.4189	0.2502	0.2875	0.3030	0.3142	0.2993	0.3314	0.3505	0.3542
SH+CNN	0.4497	0.4454	0.4585	0.4587	0.5122	0.5033	0.5162	0.5160	0.3927	0.3757	0.3780	0.3731	0.4065	0.3837	0.3853	0.3800
PCAH+CNN	0.4892	0.4914	0.4890	0.4848	0.5515	0.5514	0.5486	0.5439	0.4337	0.4185	0.4066	0.3961	0.4454	0.4207	0.4067	0.3962
ITQ+CNN	0.5606	0.5886	0.6006	0.6070	0.6429	0.6777	0.6927	0.6996	0.5137	0.5266	0.5323	0.5368	0.5328	0.5362	0.5366	0.5391
SGH+CNN	0.2575	0.2653	0.2730	0.2839	0.2773	0.2871	0.2955	0.3083	0.2129	0.2198	0.2254	0.2339	0.2675	0.2718	0.2748	0.2789
UH_BDNN	0.5572	0.5795	0.5851	0.5915	0.6388	0.6639	0.6700	0.6781	0.5080	0.5132	0.5110	0.5115	0.5188	0.5168	0.5101	0.5067
UTH	0.5389	0.5468	0.5561	0.5634	0.6192	0.6286	0.6427	0.6451	0.4856	0.4921	0.4994	0.5012	0.4961	0.4979	0.5006	0.5013
HashGAN	0.4606	0.4672	0.4711	0.4783	0.5114	0.5201	0.5263	0.5310	0.4115	0.4183	0.4214	0.4240	0.4197	0.4246	0.4293	0.4303
ODUDH	**0.7030**	**0.7469**	**0.7615**	**0.7695**	**0.7511**	**0.7998**	**0.8152**	**0.8234**	**0.5975**	**0.6200**	**0.6277**	**0.6267**	**0.6252**	**0.6321**	**0.6342**	**0.6304**

Table 1: Results on the Pascal VOC 2007. The ranking results are measured by NDCG, ACG, WMAP, and MAP@N (N=1000, i.e., the values are calculated based on the top 1000 returned neighbors). The best results for each category are shown in boldface.

Methods	MAP@5000				WMAP@5000				NDCG@5000				ACG@5000			
	12bits	24bits	36bits	48bits	12bits	24bits	36bits	48bits	12bits	24bits	36bits	48bits	12bits	24bits	36bits	48bits
LSH	0.1393	0.1494	0.1539	0.1494	0.1495	0.1602	0.1652	0.1604	0.1070	0.1136	0.1174	0.1143	0.1459	0.1543	0.1582	0.1536
SH	0.1656	0.1629	0.1641	0.1668	0.1785	0.1756	0.1768	0.1796	0.1247	0.1221	0.1235	0.1252	0.1706	0.1664	0.1677	0.1694
PCAH	0.1452	0.1464	0.1477	0.1499	0.1562	0.1575	0.1588	0.1614	0.1108	0.1110	0.1116	0.1125	0.1513	0.1516	0.1517	0.1533
ITQ	0.1356	0.1423	0.1599	0.1618	0.1431	0.1508	0.1719	0.1738	0.0996	0.1081	0.1206	0.1237	0.1326	0.1439	0.1608	0.1656
SGH	0.1681	0.1698	0.1715	0.1724	0.1807	0.1825	0.1841	0.1850	0.1280	0.1287	0.1298	0.1305	0.1696	0.1700	0.1700	0.1701
LSH+CNN	0.1621	0.1925	0.1954	0.2133	0.1750	0.2078	0.2112	0.2304	0.1233	0.1425	0.1435	0.1544	0.1657	0.1887	0.1878	0.1976
SH+CNN	0.2667	0.2805	0.2798	0.2877	0.2880	0.3041	0.3032	0.3122	0.1845	0.1981	0.1931	0.1971	0.2452	0.2460	0.2379	0.2429
PCAH+CNN	0.2991	0.3076	0.3063	0.3090	0.3236	0.3336	0.3321	0.3354	0.2215	0.2187	0.2122	0.2105	0.2803	0.2720	0.2616	0.2584
ITQ+CNN	0.3330	0.3627	0.3712	0.3781	0.3617	0.3942	0.4034	0.4123	0.2527	0.2684	0.2726	0.2765	0.3223	0.3339	0.3370	0.3415
SGH+CNN	0.1344	0.1423	0.1493	0.1575	0.1444	0.1530	0.1606	0.1697	0.1033	0.1079	0.1117	0.1152	0.1413	0.1460	0.1496	0.1527
UH_BDNN	0.3442	0.3736	0.3828	0.3960	0.3737	0.4049	0.4148	0.4289	0.2605	0.2768	0.2811	0.2876	0.3262	0.3405	0.3439	0.3500
UTH	0.3011	0.3083	0.3102	0.3138	0.3375	0.3417	0.3481	0.3495	0.2276	0.2292	0.2317	0.2343	0.2743	0.2835	0.2891	0.2931
HashGAN	0.2711	0.2790	0.2866	0.2935	0.2930	0.3052	0.3121	0.3167	0.2028	0.2091	0.2131	0.2177	0.2496	0.2528	0.2589	0.2635
ODUDH	**0.3961**	**0.4153**	**0.4252**	**0.4290**	**0.4291**	**0.4482**	**0.4577**	**0.4619**	**0.3057**	**0.3216**	**0.3269**	**0.3285**	**0.3902**	**0.4039**	**0.4088**	**0.4097**

Table 2: Results on the BMVC 2009. The ranking results are measured by NDCG, ACG, WMAP, and MAP@N (N=5000, i.e., the values are calculated based on the top 5000 returned neighbors). The best results for each category are shown in boldface.

2009 are subnet of the 81 object classes. All the weights and bias in the feature learning part and hash function learning part are learned via back-propagation algorithm. Furthermore, the weights and bias in the feature learning part are initialized as the values pre-trained in Alexnet [Krizhevsky *et al.*, 2012]. We adopt SGD with a mini-batch size of 128 as our optimization algorithm. The learning rate is initialized as 0.01. The hyper-parameters α, β in ODDUH are empirically set as 2 and 100, respectively, and will be discussed in section 4.5. And the learning rate is adjusted to one tenth of the current learning rate every one third of epoches.

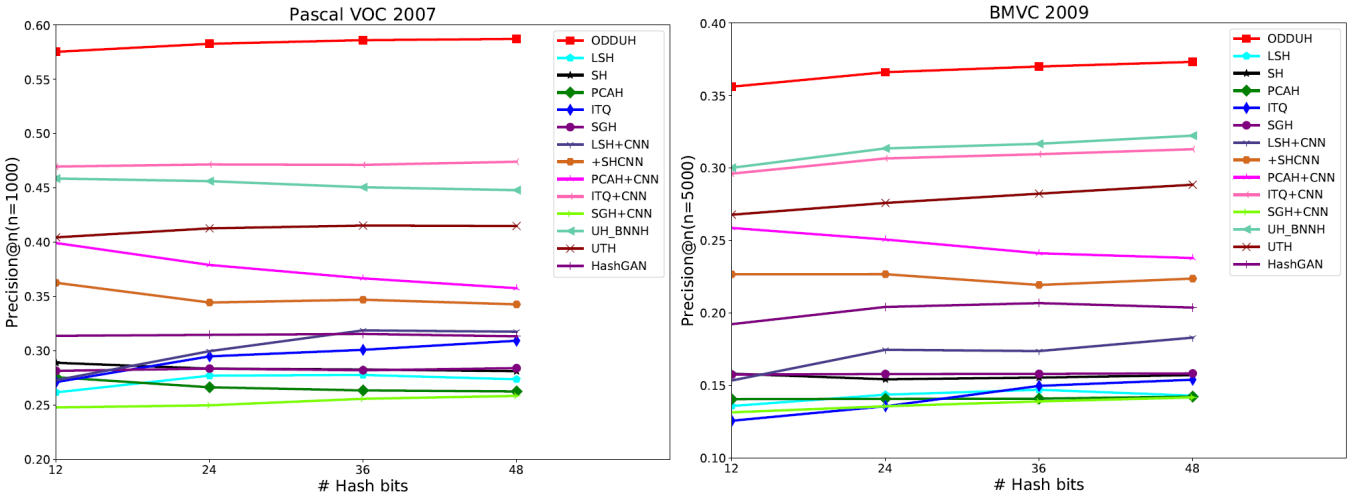
4.3 Evaluation Criterions

To verify the effectiveness of learned hash codes, we evaluate the image retrieval quality for different methods by Average Cumulative Gains (ACG) [Järvelin and Kekäläinen, 2000], Normalized Discounted Cumulative Gains (NDCG) [Järvelin and Kekäläinen, 2002], Mean Average Precision (MAP) [Baeza-Yates *et al.*, 2011], Weighted Mean Average Precision (W-MAP) [Zhao *et al.*, 2015] and Precision at top n retrieved images (Precision@k).

4.4 Experimental Results

Table 1, Table 2 and Figure 3 summarize the comparative results of different hashing methods by MAP, WMAP, NDCG ACG and Precison@n over Pascal VOC 2007 and BMVC 2009, respectively. In general, from the two tabels and Figure 3, we have two observations: (1) Our proposed method substantially outperforms the other unsupervised hashing methods for different length of hash code. For example, on Pascal VOC 2007, comparing with the best traditional competitor ITQ+CNN on 48-bits, the results of ODDUH have a relative increase of 26.8% on MAP, 17.7% on WMAP, 16.7% on NDCG, 16.9% on ACG. Moreover, on BMVC 2009, comparing with the competitor UH_BDNN on 48-bits, the results of ODDUH have a relative increase of 8.3% on MAP, 11.4% on WMAP, 14.2% on NDCG, 17.1% on ACG. In Figure 3, the precision@n curves show ODDUH can get the best performance. All these results obviously indicate that pseudo-labels mined from images are useful to improve the performance of hashing models; (2) The performance of the methods with deep features is better than the one of the methods with handcrafted features. For example, the MAP@1000 of ITQ+CNN over Pascal VOC 2007 is 0.6070 in 48 bits, and the corresponding value of ITQ is 0.3223.

In order to verify the denoisy ability of pair-wise percentage similarity on hash codes learning, an experiment is carried out over Pascal VOC 2007. As shown in Figure 4, it can be found that ODDUH with percentage similarity can get a better performance than ODDUH with two-value similarity. It verifies that pair-wise percentage similarity can decrease the harm of noisy pseudo-labels in hashing code learning procedure.



(a) (b)

Figure 3: Precision@n over (a) Pascal VOC 2007 and (b) BMVC 2009.

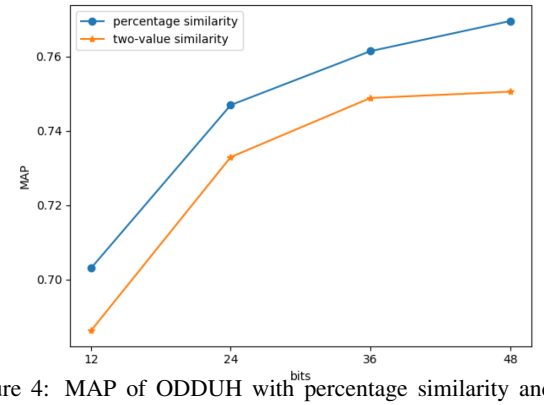


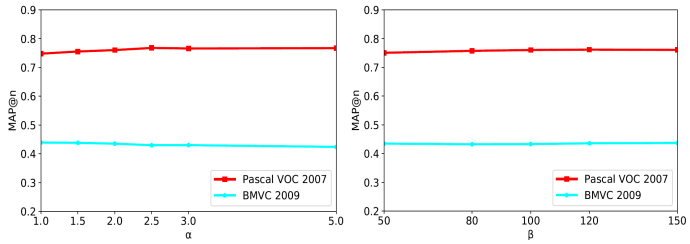
Figure 4: MAP of ODDUH with percentage similarity and two-value similarity over Pascal VOC 2007.

4.5 Sensitivity to Hyper-Parameters

Figure 5(a) shows the effect of the hyper-parameter α on 48 bits over Pascal VOC 2007 and BMVC 2009. It can be found that ODDUH is not sensitive to α on both datasets. For instance, ODDUH can achieve good performance on both datasets with $1 \leq \alpha \leq 5$. Figure 5(b) shows the effect of the hyper-parameter β on 48 bits over Pascal VOC 2007 and BMVC 2009. Also, ODDUH is not sensitive to β in a large range. For example, ODDUH can achieve good performance on both datasets with $80 \leq \beta \leq 150$. And we can also obtain similar conclusion on other length of hash codes for both hyper-Parameters α and β.

5 Conclusion

In this paper, we have proposed a novel Object Detection based Deep Unsupervised Hashing method for unlabeled data, called ODDUH. To the best of our knowledge, ODDUH is the first method which try to mine semantic 'label information' from images to guide hashing function learning. By incorporating the semantic 'label information' into the training process, the learnt hashing functions can generate high-quality similarity-preserving hash codes. Extensive experiments on two real-world public datasets have shown that the proposed ODDUH method can outperform the state-of-the-art baselines in image retrieval application.



(a) Sensitivity to hyper-parameter α over two dataset with $\beta = 100$ (b) Sensitivity to hyper-parameter β over two dataset with $\alpha = 2$

Figure 5: Sensitivity to hyper-parameters.

Acknowledgments

The work is supported by SFSMBRP(2018YFB1005100), BIGKE(No. 20160754021), NSFC (No. 61772076 and 61751201), NSFB (No. Z181100008918002), Major Project of Zhijiang Lab (No. 2019DH0ZX01), and CETC(No. w-2018018).

References

[Allan and Verbeek, 2009] Moray Allan and Jakob Verbeek. Ranking user-annotated images for multiple query terms. In *BMVC 2009-British Machine Vision Conference*, pages 20–1. BMVA Press, 2009.

[Baeza-Yates *et al.*, 2011] Ricardo Baeza-Yates, Berthier de Araújo Neto Ribeiro, et al. *Modern information retrieval*. New York: ACM Press; Harlow, England: Addison-Wesley,, 2011.

[Cao *et al.*, 2017] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and S Yu Philip. Hashnet: Deep learning to hash by continuation. In *ICCV*, pages 5609–5618, 2017.

[Datar *et al.*, 2004] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

[Do *et al.*, 2016] Thanh-Toan Do, Anh-Dzung Doan, and Ngai-Man Cheung. Learning to hash with binary deep neural network. In *European Conference on Computer Vision*, pages 219–234. Springer, 2016.

[Everingham *et al.*, 2010] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[Ghasedi Dizaji *et al.*, 2018] Kamran Ghasedi Dizaji, Feng Zheng, Najmeh Sadoughi, Yanhua Yang, Cheng Deng, and Heng Huang. Unsupervised deep generative adversarial hashing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3664–3673, 2018.

[Gionis *et al.*, 1999] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *Vldb*, volume 99, pages 518–529, 1999.

[Gong and Lazebnik, 2011] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 817–824. IEEE, 2011.

[He *et al.*, 2017] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[Huang *et al.*, 2017] Shanshan Huang, Yichao Xiong, Ya Zhang, and Jia Wang. Unsupervised triplet hashing for fast image retrieval. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*, pages 84–92. ACM, 2017.

[Järvelin and Kekäläinen, 2000] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 41–48. ACM, 2000.

[Järvelin and Kekäläinen, 2002] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

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

[Jin *et al.*, 2014] Zhongming Jin, Cheng Li, Yue Lin, and Deng Cai. Density sensitive hashing. *IEEE Trans. Cybernetics*, 44(8):1362–1371, 2014.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[Kulis *et al.*, 2009] Brian Kulis, Prateek Jain, and Kristen Grauman. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2143–2157, 2009.

[Li *et al.*, 2015] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855*, 2015.

[Lin *et al.*, 2016] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1183–1192, 2016.

[Liu *et al.*, 2016] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[Qiu *et al.*, 2017] Zhaofan Qiu, Yingwei Pan, Ting Yao, and Tao Mei. Deep semantic hashing with generative adversarial networks. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 225–234. ACM, 2017.

[Redmon and Farhadi, 2017] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017.

[Wang *et al.*, 2010] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for scalable image retrieval. 2010.

[Wang *et al.*, 2018] Dan Wang, Heyan Huang, Chi Lu, Bo-Si Feng, Liqiang Nie, Guihua Wen, and Xian-Ling Mao. Supervised deep hashing for hierarchical labeled data. pages 7388–7395, 2018.

[Weiss *et al.*, 2009] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

[Zhao *et al.*, 2015] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1556–1564, 2015.