Hindawi
Advances in Multimedia
Volume 2017, Article ID 8961091, 10 pages
https://doi.org/10.1155/2017/8961091

Research Article

Deep Binary Representation for Efficient Image Retrieval

Xuchao Lu,[1] Li Song,[1,2] Rong Xie,[1,2] Xiaokang Yang,[1,2] and Wenjun Zhang[1,2]

[1]*Institute of Image Communication and Network Engineering, Shanghai Jiao Tong University, Shanghai, China*
[2]*Future Medianet Innovation Center, Shanghai, China*

Correspondence should be addressed to Li Song; song_li@sjtu.edu.cn

Received 25 May 2017; Revised 31 July 2017; Accepted 7 September 2017; Published 12 November 2017

Academic Editor: XiangLong Liu

With the fast growing number of images uploaded every day, efficient content-based image retrieval becomes important. Hashing method, which means representing images in binary codes and using Hamming distance to judge similarity, is widely accepted for its advantage in storage and searching speed. A good binary representation method for images is the determining factor of image retrieval. In this paper, we propose a new deep hashing method for efficient image retrieval. We propose an algorithm to calculate the target hash code which indicates the relationship between images of different contents. Then the target hash code is fed to the deep network for training. Two variants of deep network, DBR and DBR-v3, are proposed for different size and scale of image database. After training, our deep network can produce hash codes with large Hamming distance for images of different contents. Experiments on standard image retrieval benchmarks show that our method outperforms other state-of-the-art methods including unsupervised, supervised, and deep hashing methods.

1. Introduction

Millions of images are uploaded and stored on the Internet every second with the rapid development of storage technique. Given a query image, how to efficiently locate a certain number of content similar images from a large database is a big challenge. Speed and accuracy need to be carefully balanced. This kind of task is content-based image retrieval (CBIR) [1–4], a technique for retrieving images by automatically derived features such as colour, texture, and shape. There are also some applications of CBIR like free-hand sketch-based image retrieval [5] whose query images are abstract and ambiguous sketches. In CBIR, derived features are not easy to store. Searching from millions and even billions of images is very time-consuming.

The binary representation of images is an emerging approach to deal with both storage and searching speed of CBIR task. This method is called hashing method, and it works in three steps. First, use a hash function to map database images (gallery images) into binary codes and store them on the storage device; the typical length is 48 bits. Then calculate the Hamming distance between the binary code of query image and stored binary codes. The images with smallest Hamming distance to the query image indicate similar content and should be retrieved. Some examples of proposed hashing methods are in [6–11].

The critical part of hashing method is the features it uses to derive the hash code. The process of all hashing method includes feature extracting; the quality of feature directly affects the retrieval accuracy. Recently, convolutional neural network (CNN) has proved its remarkable performance in tasks highly depending on feature extracting, like image classification [12], natural language processing [13], and video analysis [14]. CNN based methods outperform previous leading ones in these areas, which shows that CNN can learn robust features representing the semantic information of images. A very natural idea is to use deep learning for learning compact binary hash codes. Following semantic hashing [15], deep hashing methods using CNN show high performance in content-based image retrieval.

In this paper, we propose a new supervised deep hashing method for learning compact hash code to perform content-based image retrieval; we call it deep binary representation (DBR). This paper is an extended version of the work [16]. Our method is an end-to-end learning framework with three main steps. The first step is to generate optimal target hash

code from pointwise label information. The second step is to learn image features and hash function simultaneously through the training process of carefully designed deep network. The third step is to map image pixels to compact binary codes through a hash function and perform image retrieval. Compared to other deep hashing methods, our method has the following merits.

(1) Our deep hash network is trained with calculated target hash code. The target hash code is optimal that Hamming distance between different labels is maximized. Methods like [17] derive hash codes from a middle layer of the deep network. Our method produces hash codes from the output layer. This method is more direct and shows better performance.

(2) Our training process is pointwise; one training sample consists of one image and one target hash code. Compared to pairwise [18] and triplet methods [19, 20] whose training process needs two images or three images as one training sample, the training time is largely shortened. Our training process is a linear time algorithm, not exponential time algorithm for methods mentioned above.

(3) Our method reaches state-of-the-art performance on both small image datasets like CIFAR-10 and relatively large datasets like ImageNet. For large image datasets, we further propose an architecture based on inception-v3 net [21]; we call it DBR-v3. DBR-v3 achieves state-of-the-art performance on image retrieval of ImageNet dataset. When we apply DBR-v3 to CIFAR-10, a 15 percent performance improvement is achieved.

2. Overview of Hashing Methods

Hashing methods include data-independent methods [22] and data-dependent methods [10, 23–26]. Methods of the first category are proposed in earlier days. The most representative ones are locality-sensitive hashing (LSH) [22] and the variants of it. The hash function is not related to training data. Instead, they do random projections to map images into a feature space. The second category learns the hash function from the training data. Because of the extra information, data-dependent methods outperform data-independent ones.

Data-dependent methods can be further divided into unsupervised methods and supervised methods. Unsupervised methods include spectral hashing (SH) [23] and iterative quantization (ITQ) [26]. These methods learn hash functions from unlabelled training sets. To deal with the more complicated image database, supervised methods are proposed to learn a better hash function from label information of training images. For example, supervised hashing with kernels (KSH) [24] requires a limited amount of supervised information and achieves high-quality hashing. Minimal loss hashing (MLH) [25] is based on structured prediction with latent variables and a hinge-like loss function. And binary reconstructive embedding (BRE) [10] develops an algorithm for learning hash functions based on explicitly minimizing the reconstruction error between the original distances and the Hamming distances. Asymmetric inner-product binary coding (AIBC) [27] is a special hashing method based on asymmetric hash functions. AIBC learns two different hash functions for query images and dataset images. It can be applied to both unsupervised datasets and supervised datasets.

Hashing methods mentioned above use hand-crafted features which are not powerful enough for more complicated semantic similarity. Moreover, the feature extracting procedure is independent of hash function learning. Recently, CNN based hashing methods called deep hashing methods are proposed to issue these problems. CNN can learn more representative features over hand-crafted features. Furthermore, most deep hashing methods perform feature learning and hash function learning simultaneously and show great improvement compared to previous methods. Several deep hashing methods have been proposed and proved to have better accuracy in content-based image retrieval. For example, CNNH [18] proposes a two-stage deep hashing method. It simultaneously learns features and hash functions based on learned approximate hash codes. Deep pairwise-supervised hashing (DPSH) [28] performs learning based on pairwise labels. Reference [29] poses hash learning as a problem of regularized similarity learning and simultaneously learns hash function and image features through triplet samples. Our approach proposed in this paper outperforms the above methods.

3. Proposed Method

Our method aims to find a hash function solving the content-based image retrieval task. Given N training images $X = \{x_1, x_2, \ldots, x_N\}$ belonging to K categories, x_i is in the form of raw RGB values. Label information is noted as $L = \{l_1, l_2, \ldots, l_N\}, l_i \in \{1, 2, \ldots, K\}$. Our goal is to learn a function $H(x)$ mapping input images to compact binary codes $b_i = H(x_i)$ and $b_i \in \{0, 1\}^C$, where C stands for the hash length. The hash function $H(x)$ satisfies the following:

(1) b_i and b_j are similar in the Hamming space when $l_i = l_j$.

(2) b_i and b_j are far away in the Hamming space when $l_i \neq l_j$.

Figure 1 shows the whole flowchart of our system and Figure 2 shows the proposed network. A target hash code generation component is proposed to generate optimal hash code for training based on code length and category number. Our framework contains a CNN model as the main component. Normally the last layer of CNN is a Softmax classification layer. We replace it with a hash layer of C nodes. Since the output layer of CNN model has been changed, we need new output information to replace the labels. The hash function $H(x)$ is the trained model concatenated with a revised sgn function. Finally, we use the trained hash function to perform content-based image retrieval.

3.1. Target Hash Code Generation

3.1.1. Normal Situation. Target hash code is the mathematically optimal code set with K codewords; the Hamming distance between each codeword is maximized. We use target hash code together with raw images as the training sample to train the whole network. We hope to get a network which accepts the raw image as input and can map it to binary

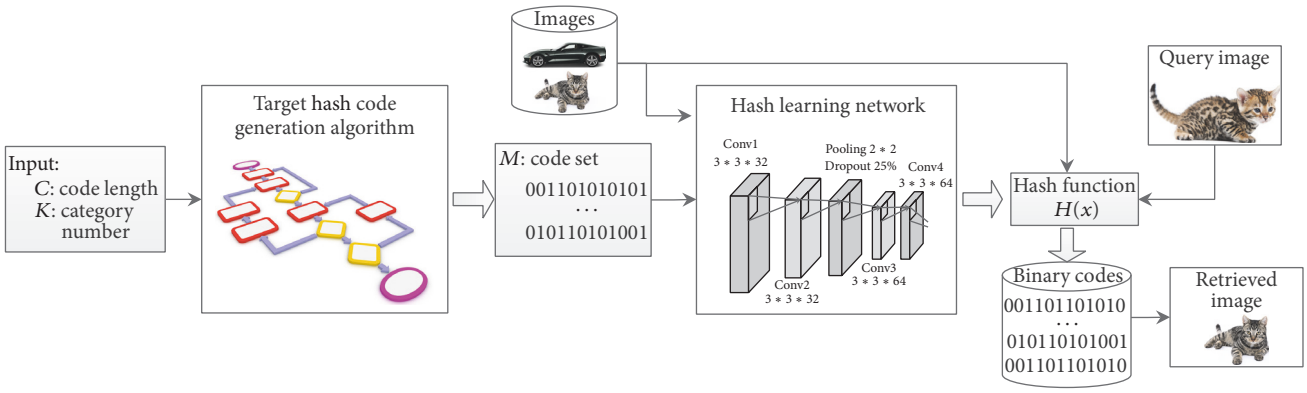

Figure 1: The process flow of our work. First, we use our proposed target hash code generation algorithm to generate optimal hash code set. Next, we use hash code set and training images to train the hash learning CNN network, which is regarded as the hash function. Finally, we use the hash function to perform content-based image retrieval.

codes close to the target hash code. The trained network, which is used as hash function $H(x)$, produces binary codes satisfying the goal. Learning the relationship between images with different labels is not our purpose. Instead, our purpose is to teach the network how to map image to a binary code. That is why we calculate the target hash codes and feed it to the network, not letting the network learn from original labels. This is the major difference between our method and others. Furthermore, the target hash code generation component makes our learning a pointwise manner. We require no pairwise inputs like [18], and the training speed is much faster. Our target hash code's function is similar to the prototype code in adaptive binary quantization (ABQ) [30]. The difference is that, in ABQ, the binary code of any data point is represented by its nearest prototype. The output binary code of the hash function lies in the prototype code set. In our method, the target hash code is only used for training. Hash function can produce binary codes not in target hash code set.

To fit the target hash code length, we replace the last layer of original CNN classification model with a fully connected layer called hash layer which has C nodes. How to generate the target hash code for images in K different labels is the main focus of this part. The following is the detailed problem description and main algorithm.

Since the training images are in K categories, our target is to find a binary code set with K codewords. The minimum Hamming distance between any two codewords should be as large as possible. In a more specific way, given binary code length C and codeword number K, we want to find a code set $M = \{m_1, m_2, \ldots, m_K\}$, $m_i \in \{0,1\}^C$, whose minimum Hamming distance is maximized. This optimization problem can first be divided into smaller jobs: given code length C and minimum Hamming distance H, find a code set with more than K code words. After that, repeat this process with larger H until no code set can be found. The last solvable H is the maximized minimum Hamming distance. The whole process is described in Algorithm 1. Please note that this optimization problem is a complicated problem with no fixed result. For different C and H, the scale of code set may not be a certain number [31]. We have proved that our algorithm is able to at least find a second optimal solution in our experiment cases. Consider a 24-bit code set for a 12-category dataset, the best solution is a code set whose minimum Hamming distance is 13 bits. Our algorithm will find one with the minimum Hamming distance of 12 bits.

Table 1: Target 12-bit hash code set for a 10-category dataset.

Label	Decimal code	Target hash code
0	504	000111111000
1	1611	011001001011
2	1652	011001110100
3	1932	011110001100
4	1971	011110110011
5	2709	101010010101
6	2730	101010101010
7	2898	101101010010
8	2925	101101101101
9	3294	110011011110

For instance, given code length $C = 12$ for a dataset with $K = 10$ categories, with our algorithm, the minimum Hamming distance $H = 6$ results in a code set M with 16 codewords. We further try $H = 7$ and it results in a set with 4 codewords, which fails to meet our need. Then we randomly choose 10 codewords from the former set $M(C = 12, H = 6)$ and the target hash code is constructed as Table 1 shows.

3.1.2. Semantically Uneven Situation. In some situations, the semantic relation between different labels is not evenly distributed. For example, image samples in a dataset are divided into 3 different categories and their labels are $L = \{l_1, l_2, l_3\} = \{\text{cat}, \text{dog}, \text{car}\}$. The images belonging to labels l_1 and l_2 are of different categories but quite similar. However, images of label l_3 are really far away from l_2 and l_3. When we input a cat as a query image, we hope to retrieve dogs before cars. The target hash code needs to be redesigned; an evenly distributed Hamming distance between each label is

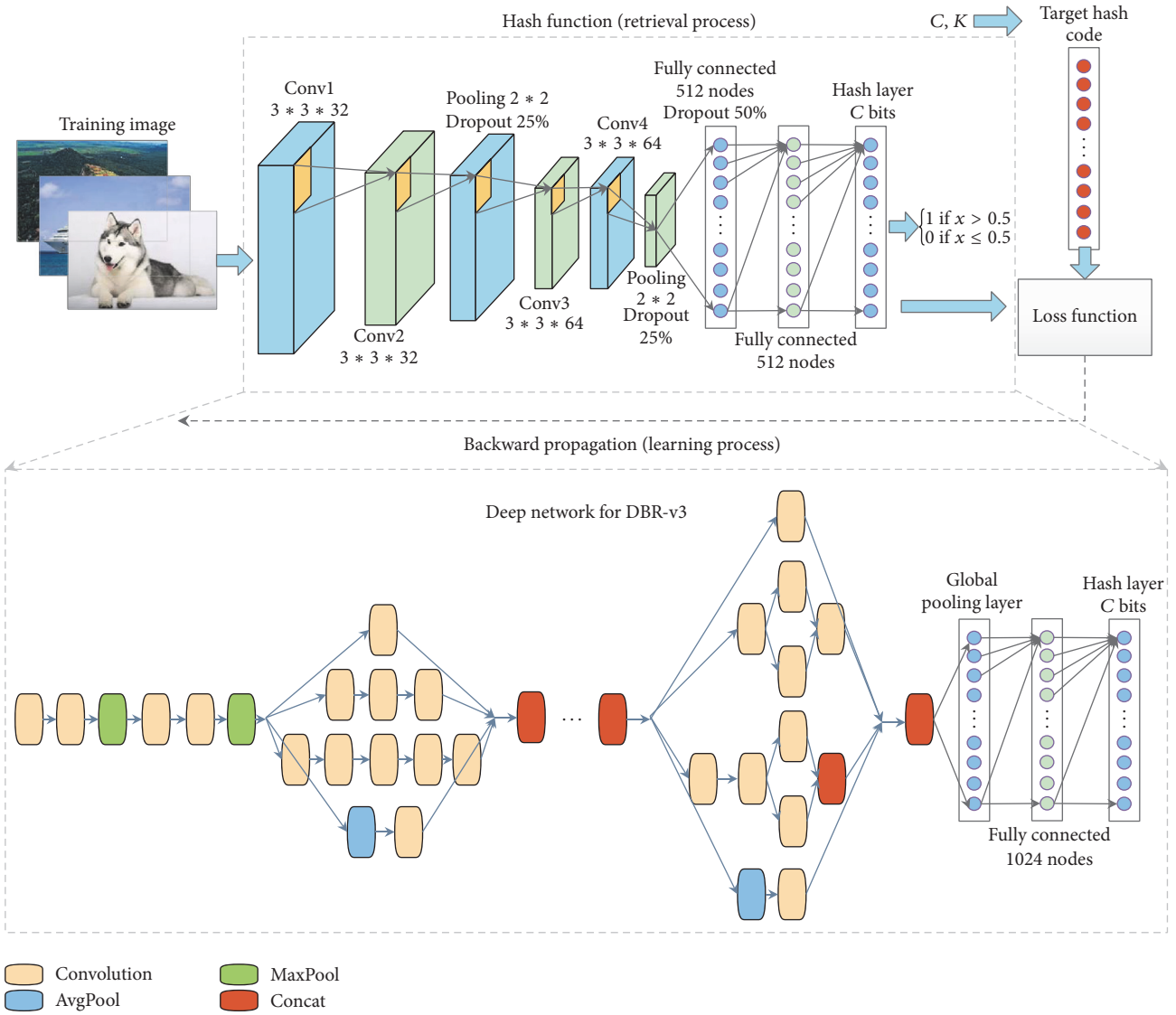

FIGURE 2: The CNN network of our proposal. The upper part is the DBR network, and the lower one is the DBR-v3 network based on inception-v3 net. First, target hash code set is generated based on hash length C and image category number K. Then deep network is trained with raw images and target hash code. Finally, image retrieval is processed with the hash function, which is the trained network concatenated with a sgn function.

not reasonable. In this example, we need a target hash code set $M = \{m_1, m_2, m_3\}$ with small Hamming distance between m_1 and m_2. In this {cat, dog, car} example, target hash code set $M = \{11001, 11010, 00000\}$ is a reasonable one since the Hamming distance between cat and dog is 2 and others are 3.

To generate such target hash code set, we need further information called semantic relation matrix S. S is a $K \times K$ matrix. Each element $s_{i,j}$ in S shows the semantic relation between label l_i and label l_j. So $s_{i,j}$ always equals $s_{j,i}$ and $s_{i,i} = 0$. A negative number means closer relation than average, for example, cat and dog. A positive number means more dissimilar relation than other labels. Zero value means the normal relation between two labels and most values should be zero. For the {cat, dog, car} example, S will be a 3×3 matrix. $s_{1,2}$ and $s_{2,1}$ are -1 and all other values are 0. To generate this kind of uneven semantic target hash code, we need a slight revise to Algorithm 1. The whole process is shown in Algorithm 2. The only difference is in line (5). When comparing the Hamming distance between the generating code and already generated code, we need to add the corresponding s of these two codes defined in semantic relation matrix S. In Algorithm 2, s means the value in S of the currently compared codes. For example, if we are generating the fifth code and comparing it to the first code in current code set, s is the value of $s_{5,1}$.

For instance, given code length $C = 12$ for a dataset with $K = 10$ categories, the semantic relation matrix S is defined as follows. Labels l_1, l_2, l_3 are very similar and their Hamming distance should be closer, so $s_{1,2} = s_{1,3} = s_{2,3} = -2$. Labels l_8, l_9, l_{10} are very dissimilar and their Hamming distance should

Input: binary code length C, number of categories K
Output: code set $\{m_1, m_2, \ldots, m_K\}$, $m_i \in \{0,1\}^C$ satisfies $\min(\text{Hamming}(m_i, m_j)) \geq H$
(1) codeset.add(0)
(2) **for** $(i = 1, 2, \ldots, 2^C - 1)$ **do**
(3) flag=0
(4) **for** j = codeset[0], codeset[1], ... **do**
(5) **if** Hamming$(i, j) < H$ **then**
(6) flag=1
(7) break
(8) **if** flag==0 **then**
(9) codeset.add(i)
(10) **return** codeset
Repeat: Perform the algorithm with $H = 1, 2, 3, \ldots$ until the length of code set is larger than K. Choose K codewords from the code set with largest H, that will be the target hash code set.

Algorithm 1: Optimal hash code generation.

Input: binary code length C, number of categories K, semantic relation matrix S. s means the corresponding value in S of the currently being compared two codes.
Output: code set $\{m_1, m_2, \ldots, m_K\}$, $m_i \in \{0,1\}^C$ satisfies $\min(\text{Hamming}(m_i, m_j)) \geq H$
(1) codeset.add(0)
(2) **for** $(i = 1, 2, \ldots, 2^C - 1)$ **do**
(3) flag=0
(4) **for** j = codeset[0], codeset[1], ... **do**
(5) **if** Hamming$(i, j) < H + s$ **then**
(6) flag=1
(7) break
(8) **if** flag==0 **then**
(9) codeset.add(i)
(10) **return** codeset
Repeat: Perform the algorithm with $H = 1, 2, 3, \ldots$ until the length of code set is larger than K. Choose K codewords from the code set with largest H, that will be the target hash code set.

Algorithm 2: Uneven semantic optimal hash code generation.

be larger, so $s_{8,9} = s_{8,10} = s_{9,10} = 2$. All other values are zero. The target hash code is constructed as Table 2 shows. The first three codes have Hamming distance 4 and the last three codes have Hamming distance 8. All other codes have the minimum Hamming distance 6. This code set satisfied the semantic relation between different labels. To the best of our knowledge, there is no dataset that includes numerical defined semantic relation between different labels. We state our algorithm here to give a solution to such semantically uneven label situation.

3.2. Learning Hash Function. With the label information $L = \{l_1, l_2, \ldots, l_N\}$ and target hash code set $M = \{m_1, m_2, \ldots, m_K\}$, we can construct our new training set T with N training samples; x_i is raw RGB value of training images and m_{l_i} is the target hash code for x_i:

$$T = \left\{\left(x_1, m_{l_1}\right), \left(x_2, m_{l_2}\right), \ldots, \left(x_N, m_{l_N}\right)\right\}. \quad (1)$$

After preparing the training samples, we build a deep network to learn to map images to hash codes. For small image datasets with the size of around $30 * 30$ pixels, we present DBR network based on relatively shallow convolutional neural networks. For large image datasets with image size of $299 * 299$, we build our network called DBR-v3 based on inception-v3 [21].

Table 2: Semantically uneven target 12-bit hash code set for a 10-category dataset: Hamming distances between codes 0, 1, and 2 are small and those between codes 7, 8, and 9 are large.

Label	Decimal code	Target hash code
0	0	000000000000
1	15	000000001111
2	51	000000110011
3	252	000011111100
4	853	001101010101
5	874	001101101010
6	1430	010110010110
7	1449	010110101001
8	2714	101010011010
9	3174	110001100110

3.2.1. Deep Network for Small Images. For small image deep network, we call our method deep binary representation (DBR). We take CIFAR-10 training as an example. We adopt a widely used simple CNN model for CIFAR-10 for fast retrieval. CNN has the powerful ability to learn image features through the concatenation of convolution layer and fully connected layer. As shown in Figure 2, we use 32, 32, 64, 64 3×3 convolution kernels for the convolution layers. 2×2 max pooling and 25% dropout are added after 2nd and 4th convolution layer. Following convolution layers are two fully connected layers with 512 nodes and a 50% dropout after the first one. All these layers are activated with ReLU function for adding nonlinearity. The hash layer is a fully connected layer with C nodes, depending on the length of target hash code. For larger C, the network can be trained to learn more features from the input image and lead to better performance. Each node implies a hidden feature of the input image. Sigmoid function ranges in $(0, 1)$ and for most cases lies in $(0, 0.1) \cup (0.9, 1)$. It is very suitable for indexing the output to binary codes.

Target hash code includes all the information needed to learn features from images, so loss function need not be specially designed; simple mean squared error (MSE) loss function works well. For training optimizer, we choose Adadelta [32] for its good balance of speed and convergence point. Without huge modifications on the CNN model, our proposed model can learn a robust hash function fast in hundreds of training epochs.

3.2.2. Deep Network for Large Images. For input images with relatively large size like $299 * 299$ pixels or similar size, we call our method DBR-v3. We build our deep network based on inception-v3 [21]. Inception net-v3 is a very deep convolutional neural network with more than 20 layers evolving from inception v1 [33]. This network achieves 21.2% top-1 and 5.6% top-5 error in ILSVRC for single frame evaluation with a computational cost of 5 billion multiply-adds per inference and with using less than 25 million parameters. We adopt the ImageNet pretrained inception-v3 model as our basic model and perform our revise and training.

We make some changes to the inception-v3 to make it fit our hash function. After the final global pooling layer, we add one fully connected layer with 1024 nodes activated with ReLU function. Following this layer is the hash layer, a fully connected layer with C nodes. The weights of newly added layers are randomly initialized and others are pretrained weights of original inception-v3. The training is a two-step process. First, we train the whole network for several epochs. After the top layers are well trained, we freeze the bottom layers and fine-tune the top 2 inception-blocks for several epochs. The loss function and training optimizer are also MSE and Adadelta [32].

This network accepts input images of $299 * 299$ pixels. For small image datasets, we use the upsampling algorithm to make images fit the network. This can make further performance improvement compared to original shallow network in Section 3.2.1. This performance improvement comes from two aspects, the power of pretrained weights and deeper networks.

3.3. Image Retrieval. After training, we combine all the components together to perform image retrieval. Our trained network accepts an input image x in raw pixels and gives an output $y \in (0, 1)^C$. To convert output y to binary hash codes $h \in \{0, 1\}^C$, we redefine the sgn function:

$$\text{sgn} = \begin{cases} 1 & \text{if } x > 0.5 \\ 0 & \text{if } x \leq 0.5. \end{cases} \tag{2}$$

Finally, we get our hash function:

$$H(x) = \text{sgn}(y) = h, \tag{3}$$

where y is the output of our proposed model.

For image retrieval, we regard training images as the image database and test images as query images. Image retrieval process searches top A most similar images from the database. Three steps are performed to do the image retrieval.

Step 1. Map N training images to hash codes $H_{db} = \{H(x_1), H(x_2), \ldots, H(x_N)\} = \{h_1, h_2, \ldots, h_N\}$.

Step 2. For each query image x_q, first calculate $h_q = H(x_q)$ and then retrieve A images by the rank of the Hamming distance (h_q, x_i); smaller Hamming distance ranks higher.

Step 3. Compare the similarities of retrieved images and the query image. Then evaluate the performance in MAP according to the result.

4. Experiments

In this part, we state our experiment settings and results. We calculate the MAP (mean average precision) of the image retrieval on different datasets and list it in Table 3. We apply our DBR method on MNIST and CIFAR-10, and DBR-v3 method on ImageNet. Furthermore, we upsample the images in CIFAR-10 and apply DBR-v3 to it. For each method, we list the time each network costs to train and to calculate the hash code of one image. The running time is listed in Table 6. We choose not to calculate the time of retrieving one image. The reason is that once the hash length is determined, the time to retrieve images according to the hash code is the same for all hashing methods. What matters is the time to map one image to the hash code. Please note that some results are missing because they are not available in corresponding paper and these methods are not totally open-source.

4.1. Results on MNIST. The MNIST dataset [34] consists of 70000 28×28 grey-scale images belonging to 10 categories of handwritten Arabic numerals from 0 to 9.

For MNIST, we use 32 convolution kernels of size 3×3 for each one of the two convolution layers. 2×2 max pooling and 25% dropout are added after the 2nd convolution layer. Following convolution layers are two fully connected layers with 128 nodes and a 50% dropout after the first fully connected layer. The last layer is the hash layer and the number of nodes is adjustable with hash length. The model

Table 3: MAP of image retrieval on MNIST, CIFAR10, and ImageNet dataset with 4 different bit lengths. We use 1000 query images and calculate the MAP within top 5000 returned neighbors in MNIST and CIFAR10 dataset. We use images from 100 random categories in ImageNet dataset and all validation images of these categories are used as query sets.

Method	MNIST (MAP)				CIFAR-10 (MAP)				ImageNet (MAP)			
	12 bits	24 bits	32 bits	48 bits	12 bits	24 bits	32 bits	48 bits	16 bits	32 bits	48 bits	64 bits
DBR-v3	—	—	—	—	*0.826*	*0.837*	*0.842*	*0.847*	*0.733*	*0.761*	*0.768*	*0.769*
DBR	*0.980*	*0.984*	*0.984*	*0.990*	*0.612*	*0.648*	*0.658*	*0.680*	—	—	—	—
HashNet	—	—	—	—	—	—	—	—	0.442	0.606	0.663	0.684
DHN	—	—	—	—	0.555	0.594	0.603	0.621	0.311	0.472	0.542	0.573
DNNH	—	—	—	—	0.552	0.566	0.558	0.581	0.290	0.461	0.530	0.565
CNNH+	0.969	0.975	0.971	0.975	0.465	0.521	0.521	0.532	—	—	—	—
CNNH	0.957	0.963	0.956	0.960	0.439	0.511	0.509	0.522	0.281	0.450	0.525	0.554
KSH	0.872	0.891	0.897	0.900	0.303	0.337	0.346	0.356	0.160	0.298	0.342	0.394
ITQ-CCA	0.659	0.694	0.714	0.726	0.264	0.282	0.288	0.295	0.266	0.436	0.548	0.576
MLH	0.472	0.666	0.652	0.654	0.182	0.195	0.207	0.211	—	—	—	—
BRE	0.515	0.593	0.613	0.634	0.159	0.181	0.193	0.196	0.063	0.253	0.330	0.358
SH	0.265	0.267	0.259	0.250	0.131	0.135	0.133	0.130	0.207	0.328	0.395	0.419
ITQ	0.388	0.436	0.422	0.429	0.162	0.169	0.172	0.175	0.326	0.462	0.517	0.552
LSH	0.187	0.209	0.235	0.243	0.121	0.126	0.120	0.120	0.101	0.235	0.312	0.360

training uses Adadelta optimizer with mean squared error loss function.

Our proposed method is compared with state-of-the-art hashing methods including data-independent method LSH [22], two unsupervised methods SH [23] and ITQ [26], four supervised methods KSH [24], MLH [25], BRE [10], and ITQ-CCA [26], and CNN based deep hashing method CNNH [18] and its variant CNNH+ [18].

We follow the experiment configurations of [18] and derive results from the same resource. We randomly select 100 images per class and total 1000 images as test query images. For the unsupervised methods, we use all the rest images as the training set. And for supervised methods including CNNH, CNNH+, and ours, we select 5000 images (500 images per class) as the training set. The whole training process lasts around 120 s for 100 epochs training on a GTX1060 6 GB graphic processing unit. It costs around 80 us to map one MNIST image to its hash code.

To evaluate the performance of retrieval, we use mean average precision (MAP). For each query image, we calculate the average precision of retrieved images. MAP is the mean value of these average precisions. Please note that, for each query image, the correctness of high ranking retrieved image counts more. The MAP result of our test is shown in Table 3; the DBR column is our method. We can find that our method outperforms other methods in grey-scale image retrieval.

4.2. Results on CIFAR-10. CIFAR-10 [35] dataset consists of 60000 32×32 images belonging to 10 categories including airplane, automobile, and bird. The layer information of CIFAR-10 implementation is stated in Section 3.2.

Other than the methods we compared in Section 4.1, we compare our method with two more CNN based deep hash methods DHN [36] and DNNH [20]. And we also follow their experiment configuration. We randomly select 100 images per class as query set. For the unsupervised methods, we use all the rest images as the training set. For supervised methods, 5000 images (500 images per class) are randomly selected as the training set. The whole training process lasts around 600 s for 300 epochs training on a GTX1060 6 GB graphic processing unit. It costs around 160 us to map one CIFAR-10 image to its hash code. Two images are considered to be similar if they have the same label. The top 12 retrieved images of two query images are shown in Figure 3 as an illustration.

Furthermore, we upsample images in CIFAR-10 to the size of $299 * 299$ and apply DBR-v3 network to it. We train the whole network for 50 epochs and perform the fine-tuning for 20 epochs. The total training time is $37\text{ s} * 50 + 51\text{ s} * 20 = 2870\text{ s}$ on a GTX1060 6 GB graphic processing unit. For DBR-v3, it costs around 3 ms to map one CIFAR-10 image to its hash code.

The MAP results are in Table 3; the result of our method is shown in columns DBR and DBR-v3. We can see that our method is better than other methods including unsupervised methods, supervised methods, and deep methods with feature learning. DBR-v3 has a big advantage over DBR. This is because the network is a lot deeper and pretrained with ImageNet. However, the training time and hash code calculating time sacrifice a lot.

We also conduct experiments on semantically uneven situations. For ten categories in CIFAR-10 dataset, we suppose that the automobile and the truck are semantically similar. We set the value of $s_{\text{truck,automobile}} = -2$ and all other values in semantic relation matrix $s_{ij} = 0$. When the query image is an automobile, we can observe that trucks will be retrieved with higher rank compared to categories other than the automobile. At the same time, Table 4 shows that the overall MAP result remains at the same level. This experiment indicates that our target hash code can have the same semantic relation between different categories.

Following [28], we further do the comparison to some deep hashing methods with different experiment settings

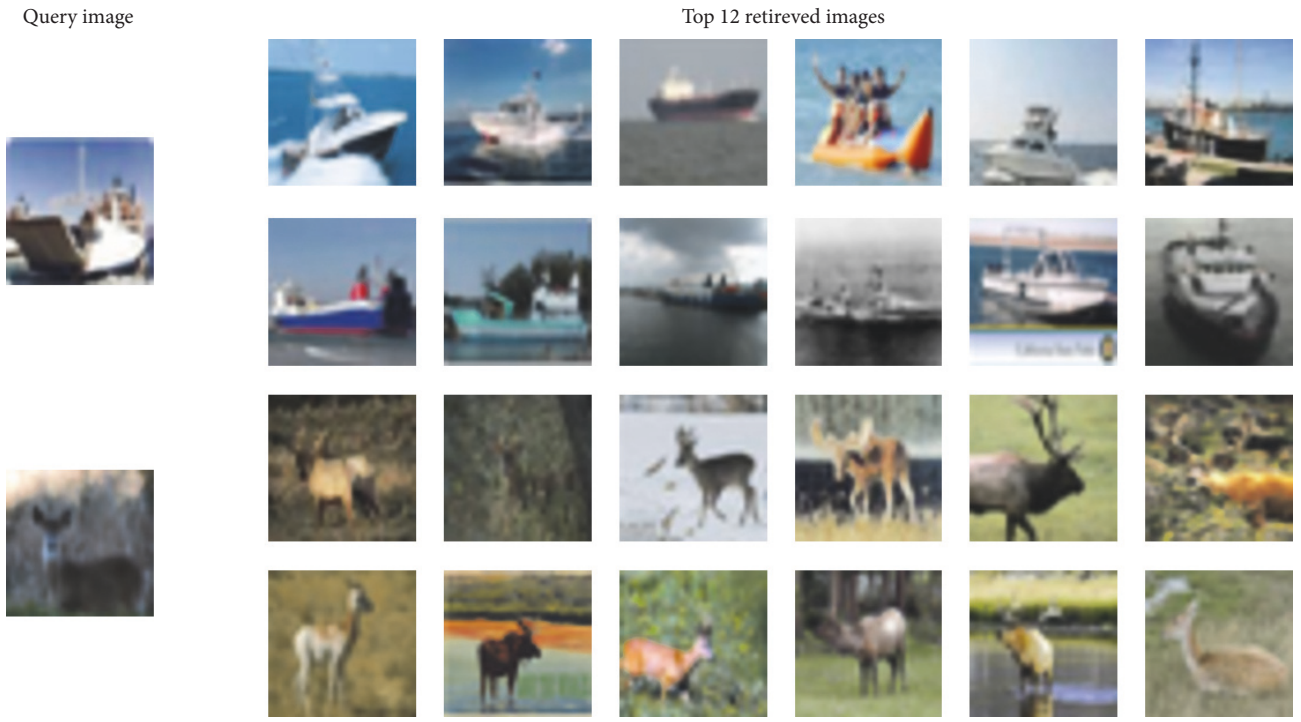

Figure 3: Top 12 retrieved images of two query image samples. The dataset is CIFAR-10 and hash length is 24 bits. As shown in the figure, the precision of high-rank retrieved images is very high.

Table 4: MAP of image retrieval on CIFAR-10 dataset with 4 different bit lengths. We use 1000 query images and calculate the MAP within top 5000 returned neighbors. The first line is the result of semantically even situation and the second is semantically uneven situation.

Method	CIFAR-10 (MAP)			
	16 bits	24 bits	32 bits	48 bits
DBR-even	0.612	0.648	0.658	0.680
DBR-uneven	0.608	0.647	0.658	0.683

Table 5: MAP of image retrieval on CIFAR-10 dataset with 4 different bit lengths. We use 10000 query images and calculate the MAP within top 50000 returned neighbors.

Method	CIFAR-10 (MAP)			
	16 bits	24 bits	32 bits	48 bits
DBR	*0.822*	*0.821*	*0.833*	*0.862*
DPSH	0.763	0.781	0.795	0.807
DRSCH	0.615	0.622	0.629	0.631
DSCH	0.609	0.613	0.617	0.620
DSRH	0.608	0.611	0.617	0.618

including DSCH, DRSCH [29], DSRH [19], and DPSH [28]; the results are directly derived from [28]. More specifically, we use 10000 test images as query set and 50000 images as the training set. The total training time is 1.5 hours for 300 epochs on a GTX1060 6 GB graphic processing unit. The MAP values are calculated according to top 50000 returned neighbours and are shown in Table 5. We can find out that our method still leads the MAP results.

4.3. Results on ImageNet. ImageNet is an image database with more than 1.2 million images in training set and more than 50 thousand images in the validation set. Each image is in one of the 1000 categories. The image size varies, and the common size is hundreds by hundreds of pixels. ImageNet is currently the largest image database for various tasks. And experiments on ImageNet show the ability to deal with large-scale high-definition images.

The network details including loss function and training optimizes are stated in Section 3.2.2. For a fair comparison, we follow the experiment settings in [37]. We randomly select 100 categories; all the images of these categories in the training set are used as training images. All the images of these categories in the validation set are used as query images. We train the whole network for 50 epochs and fine-tune the top 2 inception-blocks and hash layer for 20 epochs. The total training time is about 18 hours on a GTX1060 6 GB graphic processing unit. It costs around 3 ms to map one ImageNet image its hash code.

Our proposed method is compared to state-of-the-art hashing methods including HashNet [37] and most methods mentioned in Section 4.1. The data is derived directly from [37] and the test set is the same.

To evaluate the performance of retrieval, we use mean average precision (MAP), and the result is shown in Table 3.

Table 6: The running time of each experiment. Training time means the time to train the hash function. Hash time means the time to map one image to hash code.

Dataset	Method	Epoch	Training time	Hash time
MNIST	DBR	100	120 s	80 us
CIFAR10	DBR	300	600 s	160 us
CIFAR10	DBR-v3	50 + 20	2870 s	3 ms
ImageNet	DBR-v3	50 + 20	18 h	3 ms

The result shows that our method has a great advantage over other methods. This indicates that our DBR-v3 method can solve large-scale image retrieval for high-definition images.

5. Conclusion

In this paper, we present a novel end-to-end hash learning network for content-based image retrieval. We design the optimal target hash code for each label to feed the network with the relation between different labels. Since the target hash codes between different labels have maximized Hamming distance, the deep network can map different-category images to hash codes with significant distance. For similar images, the network tends to produce exact same hash code. The deep network is based on convolutional neural network. We design two variants of our method: (1) DBR for small images: this network trains fast and it calculates fast; with powerful clusters online training is even possible. (2) DBR-v3 based on inception-v3 net: it benefits from the powerful learning ability of inception net and performs very good on high-definition image retrieval. Finally, we do experiments on standard image retrieval benchmarks. The results show that our method outperforms the previous works.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work is supported by NSFC (61671296, 61521062, and U1611461) and the National Key Research and Development Program of China (BZ0300013).

References

[1] J. Eakins and G. Margaret, "Content-based image retrieval," 1999.

[2] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, "Content-based image retrieval at the end of the early years," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 12, pp. 1349–1380, 2000.

[3] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognition*, vol. 40, no. 1, pp. 262–282, 2007.

[4] J. Wan, D. Wang, S. C. H. Hoi et al., "Deep learning for content-based image retrieval: A comprehensive study," in *Proceedings of the 2014 ACM Conference on Multimedia, MM 2014*, pp. 157–166, usa, November 2014.

[5] L. Liu, S. Fumin, S. Yuming, L. Xianglong, and S. Ling, *Deep sketch hashing: Fast free-hand sketch-based image retrieval*, 2017, https://arxiv.org/abs/1703.05605.

[6] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, 2008.

[7] N. Mohammad and M. D. Blei, "Minimal loss hashing for compact binary codes," in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 353–360, 2011.

[8] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proceedings of the Proceeding of the 12th International Conference on Computer Vision (ICCV '09)*, pp. 2130–2137, Kyoto, Japan, October 2009.

[9] Y. Gong and S. Lazebnik, "Comparing data-dependent and data-independent embeddings for classification and ranking of Internet images," in *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011*, pp. 2633–2640, usa, June 2011.

[10] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *NIPS*, pp. 1042–1050.

[11] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '10)*, pp. 3424–3431, IEEE, San Francisco, Calif, USA, June 2010.

[12] D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 3642–3649, June 2012.

[13] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *Proceedings of the 25th International Conference on Machine Learning*, pp. 160–167, ACM, July 2008.

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and F.-F. Li, "Large-scale video classification with convolutional neural networks," in *Proceedings of the 27th IEEE Conference on Computer Vision and Pattern Recognition, (CVPR '14)*, pp. 1725–1732, Columbus, OH, USA, June 2014.

[15] R. Salakhutdinov and G. Hinton, "Semantic hashing," *International Journal of Approximate Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[16] X. Lu, L. Song, R. Xie, X. Yang, and W. Zhang, "Deep hash learning for efficient image retrieval," in *Proceedings of the IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, pp. 579–584, Hong Kong, July 2017.

[17] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proceedings of*

the IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPRW 2015, pp. 27–35, usa, June 2015.

[18] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," *AAAI Conference on Artificial Intelligence*, pp. 2156–2162, 2014.

[19] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 1556–1564, June 2015.

[20] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR'*, pp. 3270–3278, June 2015.

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 2818–2826, July 2016.

[22] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the 20th Annual Symposium on Computational Geometry (SCG '04)*, pp. 253–262, ACM, June 2004.

[23] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems (NIPS '08)*, pp. 1753–1760, Vancouver, Canada, December 2008.

[24] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 2074–2081, Providence, RI, USA, June 2012.

[25] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pp. 353–360, usa, July 2011.

[26] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.

[27] F. Shen, Y. Yang, L. Liu, W. Liu, D. Tao, and H. T. Shen, "Asymmetric Binary Coding for Image Search," *IEEE Transactions on Multimedia*, vol. 19, no. 9, pp. 2022–2032, 2017.

[28] L. Wu-Jun, W. Sheng, and K. Wang-Cheng, *Feature learning based deep supervised hashing with pairwise labels*, 2015, https://arxiv.org/abs/1511.03855.

[29] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification," *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 4766–4779, 2015.

[30] X. Liu, Z. Li, C. Deng, and D. Tao, "Distributed adaptive binary quantization for fast nearest neighbor search," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5324–5336, 2017.

[31] M. Plotkin, "Binary codes with specified minimum distance," *IRE Transactions on Information Theory*, vol. 6, no. 4, pp. 445–450, 1960.

[32] D. M. Zeiler, "Adadelta: an adaptive learning rate method," https://arxiv.org/abs/1212.5701.

[33] C. Szegedy, W. Liu, Y. Jia et al., "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '15)*, pp. 1–9, Boston, Mass, USA, June 2015.

[34] Y. LeCun, C. Cortes, and C. J. C. Burges, *The mnist database of handwritten digits*, 1994.

[35] A. Krizhevsky, N. Vinod, and H. Geoffrey, "The cifar-10 dataset," 2014.

[36] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI 2016*, pp. 2415–2421, usa, February 2016.

[37] C. Zhangjie, L. Mingsheng, W. Jianmin, and S. P. Yu, *Hashnet: Deep Learning to Hash by Continuation*, 2017, https://arxiv.org/abs/1702.00758.