Deep Supervised Hashing for Fast Image Retrieval

Haomiao Liu[1,2], Ruiping Wang[1], Shiguang Shan[1], Xilin Chen[1]
[1]Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing, 100190, China
[2]University of Chinese Academy of Sciences, Beijing, 100049, China
haomiao.liu@vipl.ict.ac.cn, {wangruiping, sgshan, xlchen}@ict.ac.cn

Abstract

*In this paper, we present a new hashing method to learn compact binary codes for highly efficient image retrieval on large-scale datasets. While the complex image appearance variations still pose a great challenge to reliable retrieval, in light of the recent progress of Convolutional Neural Networks (CNNs) in learning robust image representation on various vision tasks, this paper proposes a novel Deep Supervised Hashing (DSH) method to learn compact similarity-preserving binary code for the huge body of image data. Specifically, we devise a CNN architecture that takes pairs of images (similar/dissimilar) as training inputs and encourages the output of each image to approximate discrete values (*e.g. *+1/-1). To this end, a loss function is elaborately designed to maximize the discriminability of the output space by encoding the supervised information from the input image pairs, and simultaneously imposing regularization on the real-valued outputs to approximate the desired discrete values. For image retrieval, new-coming query images can be easily encoded by propagating through the network and then quantizing the network outputs to binary codes representation. Extensive experiments on two large scale datasets CIFAR-10 and NUS-WIDE show the promising performance of our method compared with the state-of-the-arts.*

1. Introduction

In recent years, hundreds of thousands of images are uploaded to the Internet every day, making it extremely difficult to find relevant images according to different users' request. For example, content based image retrieval retrieves images that are similar to a given query image, where "similar" may refer to visually similar or semantically similar. Suppose that both the images in the database and the query image are represented by real-valued features, the simplest way of looking for relevant images is by ranking the database images according to their distances to the query image in the feature space, and returning the closest ones. However, for a database with millions of images, which is quite common nowadays, even a linear search through the database would cost a great deal of time and memory.

To address the inefficiency of real-valued features, hashing approaches are proposed to map images to compact binary codes that approximately preserve the data structure in the original space, [27, 9, 17] for example. Since the images are represented by binary codes instead of real-valued features, the time and memory costs of searching can be greatly reduced. However, the retrieval performance of most existing hashing methods heavily depends on the features they use, which are basically extracted in an unsupervised manner, thus more suitable for dealing with the visual similarity search rather than the semantic similarity search. On the other hand, recent progress in image classification [12, 25, 8], object detection [26], face recognition [24], and many other vision tasks [18, 2] demonstrate the impressive learning power of CNNs. In these different tasks, the CNNs can be viewed as a feature extractor guided by the objective functions specifically designed for the individual tasks. The successful applications of CNNs in various tasks imply that the features learned by CNNs can well capture the underlying semantic structure of images in spite of significant appearance variations.

Inspired by the robustness of CNN features, we propose a binary code learning framework by exploiting the CNN structure, named Deep Supervised Hashing (DSH). In our method, first we devise a CNN model which takes image pairs along with labels indicating whether the two images are similar as training inputs, and produces binary codes as outputs, as shown in Figure 1. In practice, we generate image pairs online so that many more image pairs can be utilized in the training stage. The loss function is designed to pull the network outputs of similar images together and push the outputs of dissimilar ones far away, so that the learned Hamming space can well approximate the semantic structure of images. To avoid optimizing the non-differentiable loss function in Hamming space, the network

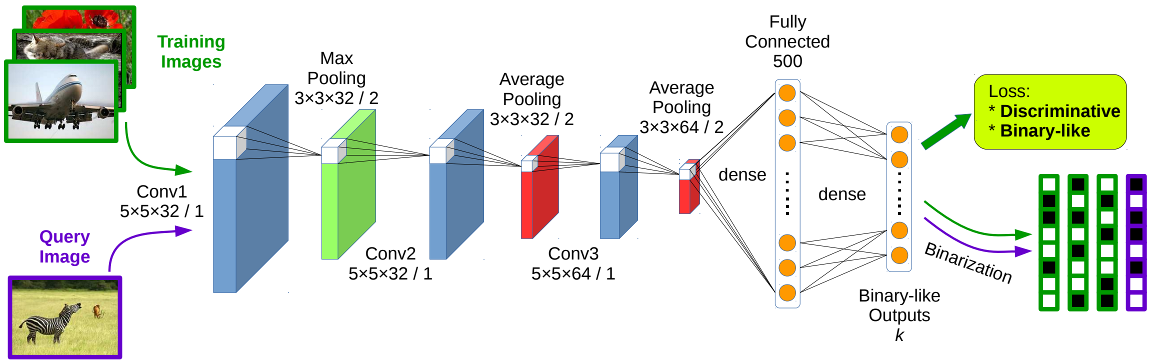

Figure 1. The network structure used in our method. The network consists of 3 convolution-pooling layers and 2 fully connected layers. The filters in convolution layers are of size 5×5 with stride 1 (32, 32, and 64 filters in the three convolution layers respectively), and pooling over 3×3 patches with stride 2. The first fully connected layer contains 500 nodes, and the second (output layer) has k (the code length) nodes. The loss function is designed to learn similarity-preserving binary-like codes by exploiting discriminability terms and a regularizer. Binary codes are obtained by quantizing the network outputs of images.

outputs are relaxed to real values, while simultaneously a regularizer is imposed to encourage the real-valued outputs to approach the desired discrete values. Under this framework, images can be easily encoded by first propagating through the network and then quantizing the network outputs to binary codes representation.

The rest of the paper is organised as follows: Section 2 discusses the related works to our method. Section 3 describes DSH in detail. Section 4 extensively evaluates the proposed method on two large scale datasets. Section 5 gives concluding remarks.

2. Related Works

Many hashing methods [4, 28, 13, 27, 6, 20, 17, 22, 16, 23, 15, 29, 31, 14] have been proposed to boost the performance of approximate nearest neighbor search due to their low time and space complexity. In the early years, researchers mainly focused on data-independent hashing methods, such as a family of methods known as Locality Sensitive Hashing (LSH) [4]. LSH methods use random projections to produce hashing bits. It has been proven theoretically that as the code length grows, the Hamming distance between two binary codes asymptotically approaches their corresponding distance in the feature space. However, LSH methods usually require long codes to achieve satisfactory performance, which demands for large amount of memory.

To produce more compact binary codes, data-dependent hashing methods are proposed. Such methods attempt to learn similarity-preserving hashing functions from a training set. These methods can be further divided into unsupervised methods and supervised (semi-supervised) methods. Unsupervised methods only make use of unlabelled training data to learn hash functions. For example, Spectral Hashing (SH) [28] minimizes the weighted Hamming distance of image pairs, where the weights are defined to be the similarity metrics of image pairs; Iterative Quantization (ITQ) [6] tries to minimize the quantization error on projected image descriptors so as to alleviate the information loss caused by the discrepancy between the real-valued feature space and the binary Hamming space.

To deal with more complicated semantic similarity, supervised methods are proposed to take advantage of label information, such as category labels. CCA-ITQ [6], which is an extension of ITQ, uses label information to find better projections for the image descriptors; Predictable Discriminative Binary Code (DBC) [22] looks for hyperplanes that separate categories with large margin as hash functions; Minimal Loss Hashing (MLH) [20] optimizes upper bound of a hinge-like loss to learn the hash functions. On the other hand, Semi-Supervised Hashing (SSH) [27] makes use of the abundant unlabelled data to regularize the hashing functions. While the above methods use linear projections as hashing functions, they can hardly deal with linearly inseparable data. To overcome this limitation, Supervised Hashing with Kernels (KSH) [17] and Binary Reconstructive Embedding (BRE) [13] are proposed to learn similarity-preserving hashing functions in kernel space; Deep Hashing (DH) [3] exploits a non-linear deep network to produce binary codes. Most hashing methods relax the binary codes to real-values in optimization and quantize the model outputs to produce binary codes. However, there is no guarantee that the optimal real-valued codes are still optimal after quantization. Methods such as Discrete Graph Hashing (DGH) [16] and Supervised Discrete Hashing (SDH) [23] are proposed to directly optimize the binary codes to overcome the shortcomings of relaxation, and achieves improved retrieval performance.

While the aforementioned hashing methods have certainly achieved success to some extent, they all use hand-crafted features, which cannot capture the semantic information beneath the drastic appearance variations in real-world data and thus limit the retrieval accuracy of the learned binary codes. To tackle this issue, most recently, several CNN-based hashing methods [31, 14, 29, 15, 30] are proposed to learn image representations together with binary codes using the promising CNNs. [31, 14, 30] enforce the network to learn binary-like outputs that preserve the semantic relations of image-triplets; [29] trains a CNN to fit the binary codes computed from the pairwise similarity matrix; [15] trains the model with a binary-like hidden layer as features for image classification tasks. By coupling image feature extraction and binary code learning, these methods have shown greatly improved retrieval accuracy. Nevertheless, there still exist some shortcomings with the training objectives of these methods that limit their practical retrieval performance, as will detailed in our experiments. In addition, the non-linear activations they employ to approximate the quantization step operate at the cost of possibly slowing down the network training [12].

3. Approach

Our goal is to learn compact binary codes for images such that: (a) similar images should be encoded to similar binary codes in Hamming space, and vice versa; (b) the binary codes could be computed efficiently.

Although many hashing methods have been proposed to learn similarity-preserving binary codes, they suffer from the limitations of either hand-crafted features or linear projections. The powerful non-linear models known as CNNs have facilitated the recent successes in computer vision community on various tasks. To this end, we propose to use the CNN illustrated in Figure 1 to learn discriminative image representations and compact binary codes simultaneously, which can break out the limitations of both hand-crafted features and linear models. Our method first trains the CNN using image pairs and the corresponding similarity labels. Here the loss function is elaborately designed to learn similarity-preserving binary-like image representations. Then the CNN outputs are quantized to generate binary codes for new-coming images.

3.1. Loss Function

Let Ω be the RGB space, our goal is to learn a mapping from Ω to k-bit binary code: $\mathscr{F} : \Omega \rightarrow \{+1, -1\}^k$, such that similar (either in terms of visually similar or semantically similar) images are encoded to similar binary codes. For this purpose, the codes of similar images should be as close as possible, while the codes of dissimilar images being far away. Based on this objective, the loss function is naturally designed to pull the codes of similar images together, and push the codes of dissimilar images away from each other.

Specifically, for a pair of images $I_1, I_2 \in \Omega$ and the corresponding binary network outputs $\mathbf{b}_1, \mathbf{b}_2 \in \{+1, -1\}^k$, we define $y = 0$ if they are similar, and $y = 1$ otherwise. The loss with respect to the pair of images is defined as:

$$\begin{aligned} L(\mathbf{b}_1, \mathbf{b}_2, y) = & \frac{1}{2}(1-y)D_h(\mathbf{b}_1, \mathbf{b}_2) \\ & + \frac{1}{2}y \ \max(m - D_h(\mathbf{b}_1, \mathbf{b}_2),\ 0) \\ & s.t. \ \mathbf{b}_j \in \{+1, -1\}^k,\ j \in \{1, 2\} \end{aligned} \quad (1)$$

where $D_h(\cdot\ ,\ \cdot)$ denotes the Hamming distance between two binary vectors, and $m > 0$ is a margin threshold parameter. The first term punishes similar images mapped to different binary codes, and the second term punishes dissimilar images mapped to close binary codes when their Hamming distance falls below the margin threshold m. Here it is worth noting that to avoid collapsed solution, our loss function takes a contrastive loss form as [7] where only those dissimilar pairs having their distance within a radius are eligible to contribute to the loss function.

Suppose that there are N training pairs randomly selected from the training images $\{(I_{i,1}, I_{i,2}, y_i)|i = 1, ..., N\}$, our goal is to minimize the overall loss function:

$$\begin{aligned} & \mathcal{L} = \sum_{i=1}^{N} L(\mathbf{b}_{i,1}, \mathbf{b}_{i,2}, y_i) \\ & s.t. \ \mathbf{b}_{i,j} \in \{+1, -1\}^k,\ i \in \{1, ..., N\},\ j \in \{1, 2\} \end{aligned} \quad (2)$$

3.2. Relaxation

It would be preferable if one can directly optimize Eqn.(2), however it is infeasible because the binary constraints on $\mathbf{b}_{i,j}$ requires thresholding the network outputs (*e.g.* with signum function), and will make it intractable to train the network with back propagation algorithm. Some recent works [23, 16] propose to directly optimize the binary codes, however, due to the memory limitation, CNN models can only be trained with mini-batches, and the optimality of the produced binary codes is questionable when the batch size is very small compared to the whole training set.

On the other hand, if one totally ignores the binary constraints, it would result in suboptimal binary codes due to the discrepancy between the Euclidean space and the Hamming space. A commonly used relaxation scheme is to utilize *sigmoid* or *tanh* function to approximate the thresholding procedure. Nevertheless, working with such non-linear functions would inevitably slow down or even restrain the convergence of the network [12]. To overcome such limitation, in this work we propose to impose a regularizer on the real-valued network outputs to approach the desired discrete values (+1/-1). To be specific, we replace the Hamming distance in Eqn.(1) by Euclidean distance, and impose an additional regularizer to replace the binary constraints, then

Eqn.(1) is rewritten as:

$$
\begin{aligned}
L_r(\mathbf{b}_1, \mathbf{b}_2, y) =& \frac{1}{2}(1-y)||\mathbf{b}_1 - \mathbf{b}_2||_2^2 \\
&+\frac{1}{2}y \ \max(m - ||\mathbf{b}_1 - \mathbf{b}_2||_2^2, \ 0) \\
&+\alpha(|| \ |\mathbf{b}_1| - \mathbf{1}||_1 + || \ |\mathbf{b}_2| - \mathbf{1}||_1)
\end{aligned}
\tag{3}
$$

where the subscript r denotes the relaxed loss function, $\mathbf{1}$ is a vector of all ones, $||\cdot||_1$ is the L1-norm of vector, $|\cdot|$ is the element-wise absolute value operation, and α is a weighting parameter that controls the strength of the regularizer.

Here we use L2-norm to measure the distance between network outputs because the subgradients produced by lower-order norms treat the image pairs with different distances equally and thus make no use of the information involved in different distance magnitudes. While higher-order norms are also feasible, more computations will be incurred accordingly at the same time. As for the regularizer, L1-norm is chosen rather than higher-order norms for its much less computational cost, which can favorably accelerate the training process.

By substituting Eqn.(3) into Eqn.(2), we rewrite the relaxed overall loss function as follows:

$$
\begin{aligned}
\mathcal{L}_r = \sum_{i=1}^{N} \{ & \frac{1}{2}(1-y_i)||\mathbf{b}_{i,1} - \mathbf{b}_{i,2}||_2^2 \\
&+ \frac{1}{2}y_i \ \max(m - ||\mathbf{b}_{i,1} - \mathbf{b}_{i,2}||_2^2, \ 0) \\
&+ \alpha(|| \ |\mathbf{b}_{i,1}| - \mathbf{1}||_1 + || \ |\mathbf{b}_{i,2}| - \mathbf{1}||_1)\}
\end{aligned}
\tag{4}
$$

With this objective function, the network is trained using back-propagation algorithm with mini-batch gradient descent method. To do so, the gradients of Eqn.(4) w.r.t. $\mathbf{b}_{i,j}, \forall i, j$ need to be computed. Since the *max* operation and the absolute value operation in the objective function is non-differentiable at some certain points, we use subgradients instead, and define the subgradients to be $\mathbf{1}$ at such points. The subgradients of the first two terms of Eqn.(4) and that of the third term (*i.e.* the regularizer) are respectively written as:

$$
\begin{aligned}
&\frac{\partial Term \ 1}{\partial \mathbf{b}_{i,j}} = (-1)^{j+1}(1-y_i)(\mathbf{b}_{i,1} - \mathbf{b}_{i,2}) \\
&\frac{\partial Term \ 2}{\partial \mathbf{b}_{i,j}} = \begin{cases} (-1)^j y_i(\mathbf{b}_{i,1} - \mathbf{b}_{i,2}), & ||\mathbf{b}_{i,1} - \mathbf{b}_{i,2}||_2^2 < m \\ \mathbf{0} & , otherwise \end{cases} \\
&\frac{\partial Regularizer}{\partial \mathbf{b}_{i,j}} = \alpha\delta(\mathbf{b}_{i,j})
\end{aligned}
\tag{5}
$$

where

$$
\delta(x) = \begin{cases} 1, & -1 \leq x \leq 0 \ or \ x \geq 1 \\ -1, & otherwise \end{cases}
\tag{6}
$$

is applied element-wisely. With the computed subgradients over mini-batches, the rest of the back-propagation can be done in standard manner.

Discussion: With such a framework, the binary codes of images are easily obtained with $sign(\mathbf{b})$. Note that unlike existing CNN-based hashing methods [29, 15, 14, 31, 30], our method does not use saturating non-linearities, *e.g. tanh* or *sigmoid*, to approximate the quantization step because these nonlinearities are likely to slow down the training process [12]. Experiments in Section 4.2 will validate the advantage of the regularizer over saturating nonlinearities.

3.3. Implementation details

Network parameters: Our DSH method is implemented with Caffe[1] [10]. The network structure is illustrated in Figure 1, which consists of three convolution-pooling layers followed by two fully connected layers. The convolution layers use 32, 32, and 64 5×5 filters with stride 1 respectively, and the pooling is performed over 3×3 windows with stride 2. The first fully connected layer contains 500 nodes, and the second contains k nodes, where k is the length of binary code. All the convolution layers and the first fully connected layer are equipped with the ReLU [19].

The weight layers are initialized with "Xavier" initialization [5]. During training, the batch size is set to 200, momentum to 0.9, and weight decay to 0.004. The initial learning rate is set to 10^{-3} and decreases by 40% after every 20,000 iterations (150,000 iterations in total). The margin m in Eqn.(4) is heuristically set to $m = 2k$ to encourage the codes of dissimilar images to differ in no less than $\frac{k}{2}$ bits.

Training methodology: An intuitive way to train the network is to use the Siamese structure [7] and generate image pairs offline. However, with such a scheme, processing n images could only produce $\frac{n}{2}$ valid image pairs, and storing the image pairs would be very space intensive. To make better use of computational resources and storage space, we propose to generate image pairs online by exploiting all the unique pairs in each mini-batch. To cover those image pairs across batches, in each iteration the training images are randomly selected from the whole training set. By doing so, our method alleviates the need to store the whole pair-wise similarity matrix, thus being scalable to large-scale datasets.

Moreover, to learn models corresponding to different code lengths, if one chooses to train each model from scratch, it would be severely wasteful since the preceding layers can be shared by these models. Besides, as the code length grows, the model would contain more parameters in the output layer, and thus gets prone to overfitting. To overcome such limitations, we propose to first train a network with a few nodes in the output layer, and then finetune it to obtain the target model with the desired code length.

[1]The source code of our DSH with running samples are available at http://vipl.ict.ac.cn/resources/codes.

Models	CIFAR-10	NUS-WIDE
Regularizer-α-0	0.5497	0.5076
Regularizer-α-0.001	0.6100	0.5341
Regularizer-α-0.01	0.6157	0.5483
Regularizer-α-0.1	0.4337	0.4493
Sigmoid-m-6	0.1451	0.4876
Sigmoid-m-3	0.2812	0.5067
Sigmoid-m-2	0.4788	0.4838
Sigmoid-m-1	0.2196	0.4638

Table 1. Retrieval performance (mAP) of models under different settings of α, relaxation, and m. The results are obtained with 12-bit binary codes.

4. Experiments

4.1. Datasets and Evaluation Metrics

We verify the effectiveness of our proposed method and compare with other state-of-the-art methods on two widely used datasets: (1) **CIFAR-10** [11]. This dataset consists of 60,000 32×32 images belonging to 10 mutually exclusive categories (6,000 images per category). The images are directly used as input for those competing CNN-based methods as well as our DSH. For conventional hashing methods, the images are represented by 512-D GIST descriptors [21] following [17, 29]. (2) **NUS-WIDE** [1]. This dataset contains 269,648 images collected from Flickr. The associations between images and 81 concepts are manually annotated. Following [17, 29], we use the images associated with the 21 most frequent concepts, where each of these concepts associates with at least 5,000 images, resulting in a total of 195,834 images. The images are warped to 64×64 before inputting to the CNN-based methods. For conventional hashing methods, images are represented by the provided 225-D normalized block-wise color moments features.

In our experiments, similarity labels are defined by semantic-level labels. For CIFAR-10, images from the same category are considered semantically similar, and vice versa. The officially provided train/test split was used for experiments, namely, 50,000 images for training the models and 10,000 images for evaluating. For NUS-WIDE, if two images share at least one positive label, they are considered similar, and dissimilar otherwise. We randomly sampled 10,000 images to form the test query set, and used the rest as training set.

Following previous works, the evaluation metrics used are: the mean Average Precision (mAP) for different code lengths, precision-recall curves (48-bit), and mean precision within Hamming radius 2 for different code lengths.

4.2. Evaluation of the Regularizer

In this part, we validate the effectiveness of the proposed regularizer, and compare it with the standard relaxation scheme used in existing CNN-based hashing methods [29, 3, 14]. Without loss of generality, we only test the case when $k = 12$, and set $m = 24$ in our DSH according to Section 3.3. The sigmoid relaxed models were trained almost the same as ours except for using sigmoid function as the activation of the output layer and setting $\alpha = 0$. We test these models with $m = \{1, 2, 3, 6\}$ (note that the maximum distance between network outputs of these models is k).

The retrieval mAP of different models are listed in Table 1. Figure 2 shows the distribution of network outputs on the test set of CIFAR-10 under different settings (more results are provided in supplementary materials). We make three observations from the comparison results: **First**, without regularization ($\alpha = 0$), the network outputs concentrate on the quantization threshold 0 (Fig.2a), thus it is likely that neighboring points in the output space are quantized to very different binary codes; **Second**, imposing the regularizer ($\alpha = \{0.001, 0.01, 0.1\}$, Fig.2b,c,d) can reduce the discrepancy between the real-valued output space and the Hamming space, and the retrieval performances can be improved significantly when setting α under a reasonable range (*e.g.* [0.001, 0.01]); **Third**, with proper settings of m, the sigmoid relaxed model can learn binary-like outputs (Fig.2e,f,g). Nonetheless, the retrieval performances of such codes are much inferior to our best-performing ones and are sensitive to m. Increasing the number of training iterations and carefully tuning m might improve the performance of the sigmoid relaxed models, however, it would take much more time to obtain a satisfactory model. Based on the above observations, we empirically set $\alpha = 0.01$ in the following experiments.

4.3. Online vs. Offline Image Pair Generation

This part compares the convergence behavior of our online image pair generating scheme against the alternative Siamese scheme, as described in Section 3.3. Both schemes employed the same network structure and hyperparameters as detailed in Section 3.3 ($k = 12$, $m = 24$). Due to limited storage space, 10 million image pairs were generated offline for the Siamese scheme, and the learning rate policy was tuned accordingly. For fair comparison, we input the same number of images to both schemes in each iteration (200 images for our online scheme and 100 image pairs for the alternative Siamese scheme). Since the computations mainly take place in the convolution-pooling layers, the computation costs of the two schemes are approximately the same.

Figure 3 shows the training loss against the number of iterations on both datasets. As can be seen, our online training scheme converges much faster than the Siamese alternative, since our online scheme has the capacity to utilize much more image pairs in each iteration, which offers more information about the semantic relations between different images. Besides, by sampling from the whole training set

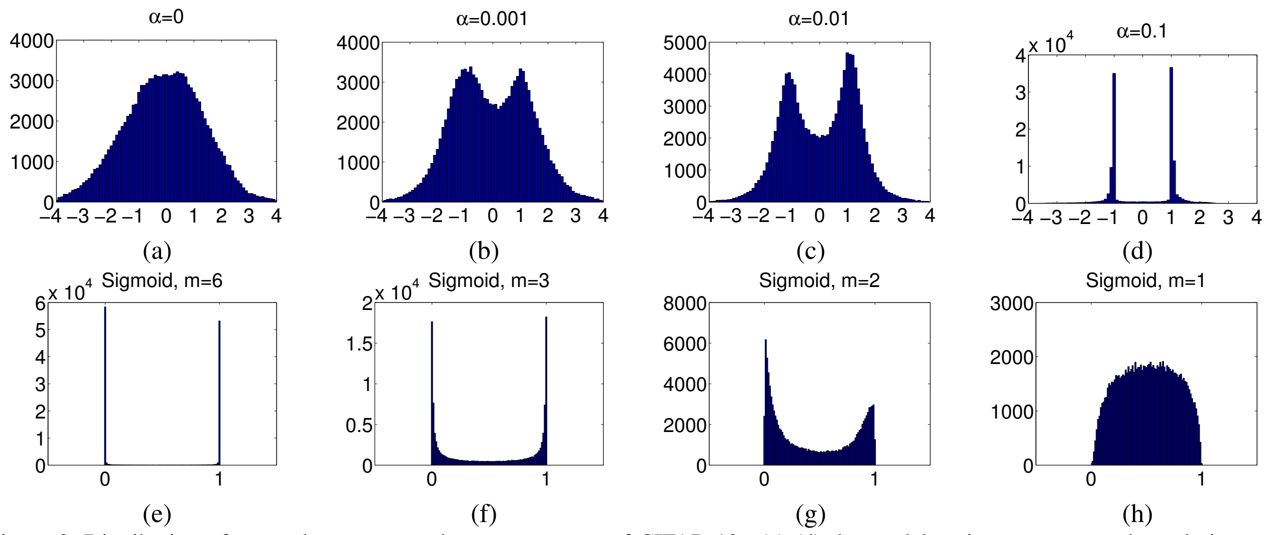

Figure 2. Distribution of network outputs on the test query set of CIFAR-10. (a)-(d) the models using our proposed regularizer under different settings of α, (e)-(h) the sigmoid relaxed models under different settings of m.

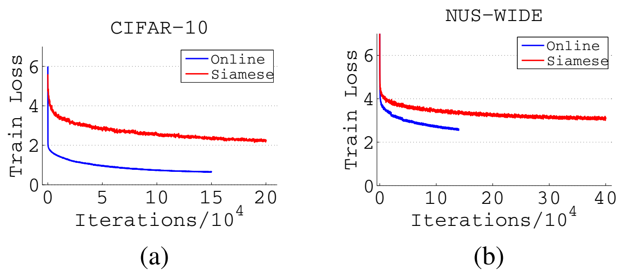

Figure 3. Comparison of training loss between our online image pair generating scheme and the Siamese alternative. The results on CIFAR-10 and NUS-WIDE are shown in (a) & (b) respectively.

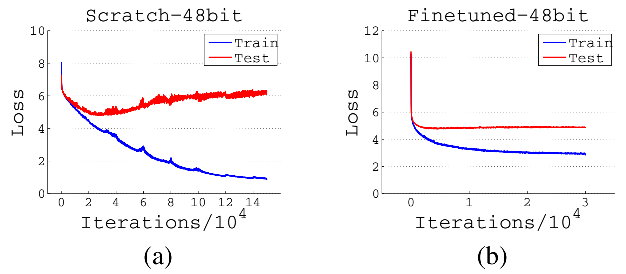

Figure 4. Comparison of (a) the model trained from scratch and (b) the finetuned model in terms of training/test loss, on the CIFAR-10 dataset.

	Code Length	CIFAR-10	NUS-WIDE
Trained From Scratch	12	0.6157	0.5483
	24	0.6524	0.5543
	36	0.6433	0.5229
	48	0.6213	0.4896
Finetuned	24	0.6512	0.5513
	36	0.6607	0.5582
	48	0.6755	0.5621

Table 2. Comparison of retrieval performance (mAP) of the models trained from scratch and the finetuned models.

in each iteration, our scheme can make use of more image pairs than the offline generated 10 million pairs for Siamese, and thus satisfactorily converges to a lower loss.

4.4. Finetuning vs. Training From Scratch

As mentioned in Section 3.3, if the last fully connected layer contains a large number of nodes, training the model from scratch may lead to overfitting. To get a clear understanding of the situation, in this part, we compare the models finetuned from a pretrained network against the models trained from scratch. Specifically, We first trained four models that produces $\{12, 24, 36, 48\}$-bit binary codes respectively (the first four rows in Table 2). Then we replaced the last fully connected layer of the 12-bit model with a larger one, and finetuned it to get another group of $\{24, 36, 48\}$-bit models (the last three rows in Table 2).

For finetuning, the learning rate was set to 10^{-3} for the last fully connected layer and 10^{-4} for the preceding layers, and decreased by a factor of 0.6 after every 4,000 iterations. The model was trained with 30,000 iterations in total.

The retrieval mAPs on both datasets are listed in Table 2. It can be found that as the code length grows, the retrieval performance of finetuned models consistently improves, while the performance of models trained from scratch falls, especially on the NUS-WIDE dataset with a large drop. To take a closer look at the situation, we analyze the training/test loss on two example models, namely, the 48-bit model trained from scratch, and the finetuned 48-bit model. Figure 4 shows the loss against the number of iterations for the two models on CIFAR-10. It is clear that on the first model (trained from scratch), the training loss keeps decreasing, while the test loss decreases as expected at first but increases after about 30,000 iterations, indicating overfitting on the training set. As comparison, on the second model (finetuned), the test loss decreases at first and then favorably stablizes after only a few thousand iterations. Such observations suggest that the different models with various code length can share those preceding layers to reduce training cost as well as to alleviate overfitting. For more results please refer to supplementary materials.

Moreover, we investigate network ensembles, which are widely used in classification tasks [12, 25, 8], for retrieval problem. Specifically, we trained four 12-bit models with different random initializations, and concatenated the quantized network outputs as binary codes. Under the same code length, the ensemble codes further improve the retrieval performance of the finetuned codes by up to 0.04 in mAP, verifying the effectiveness of network ensembles in retrieval task (the details are provided in supplementary materials). One possible explanation is that the multiple networks can capture complementary image characteristics due to random initialization. Nevertheless, since exploiting network ensembles leads to multiple times of training cost, we adopt the finetuned models in the following experiments for efficiency consideration.

4.5. Comparison with the State-of-the-art

Comparative methods: We compare our method with LSH [4], SH [28], ITQ [6], CCA-ITQ [6], MLH [20], BRE [13], KSH [17], CNNH [29], DLBHC [15], and DNNH [14]. These methods were all implemented using source codes provided by the authors except for DNNH[2]. For fair comparison, all the CNN-based methods, including CNNH, DLBHC, DNNH, and DSH, used the same network structure, as described in Section 3.3. Note that while more complicated network structures can be also feasible, we chose to work with a relatively simple one for fast evaluation.

Training set: We aim to use the whole training data to train models for all methods if possible. However, due to the huge amount of memory demanded by MLH, KSH and CNNH ($O(N^2)$, where N is the number of training images), in our experiments, we randomly selected a 20K subset from each dataset to train models for these three methods, which costs more than 10GB of memory.

Parameter settings: The parameters of those comparative methods were all set based on the authors' suggestions in the original publications. In particular, we found the divide-and-encode structure devised in DNNH [14] largely degraded the retrieval mAP on CIFAR-10 (about 0.07) and brought marginal improvement on NUS-WIDE (0.01 $\sim$ 0.03) in our experiments, thus we report the performance of the fully connected version for simplicity.

[2]Since the source code of DNNH is not publicly available, we used our own implementation of this method for experiments.

Results: The comparisons of our method against the others are shown in Table 3 and Figure 5. In general, those CNN-based methods outperform the conventional hash learning methods on both datasets by a large margin, validating the advantage of learning image representations over using hand-crafted features. Moreover, we investigate some conventional hashing methods trained with CNN features, although the performances were significantly improved, they were still inferior to our DSH, suggesting that our end-to-end learning scheme is advantageous (the details are provided in the supplementary materials).

Among the CNN-based methods, it is observed that our DSH yields the highest accuracy in most cases. The performance gaps between these methods mainly come from the differences in their training objectives: CNNH trains the model to fit the pre-computed discriminative binary codes. However, as the binary code generation and the network learning are isolated, a mismatch exists between the two stages; DLBHC trains the model with a binary-like hidden layer as features for classification tasks , thus encoding dissimilar images to similar binary codes would not be punished as long as the classification accuracy is unaffected; While DNNH uses triplet-based constraints (rather than the pairwise constraints we adopt) to describe more complex semantic relations, training its network becomes more difficult, due to the sigmoid non-linearity and the parameterized piece-wise threshold function used in the output layer. As a result, DNNH performs inferior to our DSH method, especially on CIFAR-10, where the triplet-based constraints cannot provide more information than the pairwise ones since the images only have category labels (some real retrieval cases are provided in the supplementary materials).

4.6. Comparison of Encoding Time

In real-world applications, generating binary codes for new-coming images should be fast. In this part, we compare the encoding time of our DSH method and 7 other supervised hashing methods: CCA-ITQ [6], MLH [20], BRE [13], KSH [17], CNNH [29], DLBHC [15], and DNNH [14], including the linear and non-linear conventional hashing methods along with the state-of-the-art CNN-based methods. For thorough comparison, we report the encoding time of CNN-based methods both on CPU and GPU, and the feature extraction time for conventional hashing methods (using the publicly available code of GIST feature extraction [21]). Since we used the authors' provided features for NUS-WIDE and only extracted features for CIFAR-10, all comparisons were conducted on CIFAR-10. Without loss of generality, we only report the timings of 24-bit and 48-bit codes. The binary codes of all CNN-based methods were generated with the same version of Caffe. The experiments were carried out on a PC with Intel i7-4770, 32GB RAM, and NVIDIA Titan Black with CUDA-7.0 and cuDnn v3.0.

Method	CIFAR-10				NUS-WIDE			
	12-bit	24-bit	36-bit	48-bit	12-bit	24-bit	36-bit	48-bit
LSH [4]	0.1277	0.1367	0.1407	0.1492	0.3329	0.3392	0.3450	0.3474
SH [28]	0.1319	0.1278	0.1364	0.1320	0.3401	0.3374	0.3343	0.3332
ITQ [6]	0.1080	0.1088	0.1117	0.1184	0.3425	0.3464	0.3522	0.3576
CCA-ITQ [6]	0.1653	0.1960	0.2085	0.2176	0.3874	0.3977	0.4146	0.4188
MLH [20]	0.1844	0.1994	0.2053	0.2094	0.3829	0.3930	0.3959	0.3990
BRE [13]	0.1589	0.1632	0.1697	0.1717	0.3556	0.3581	0.3549	0.3592
KSH [17]	0.2948	0.3723	0.4019	0.4167	0.4331	0.4592	0.4659	0.4692
CNNH [29]	0.5425	0.5604	0.5640	0.5574	0.4315	0.4358	0.4451	0.4332
DLBHC [15]	0.5503	0.5803	0.5778	0.5885	0.4663	0.4728	0.4921	0.4916
DNNH [14]	0.5708	0.5875	0.5899	0.5904	0.5471	0.5367	0.5258	0.5248
DSH	**0.6157**	**0.6512**	**0.6607**	**0.6755**	**0.5483**	**0.5513**	**0.5582**	**0.5621**

Table 3. Comparison of retrieval mAP of our DSH method and the other hashing methods on CIFAR-10 and NUS-WIDE.

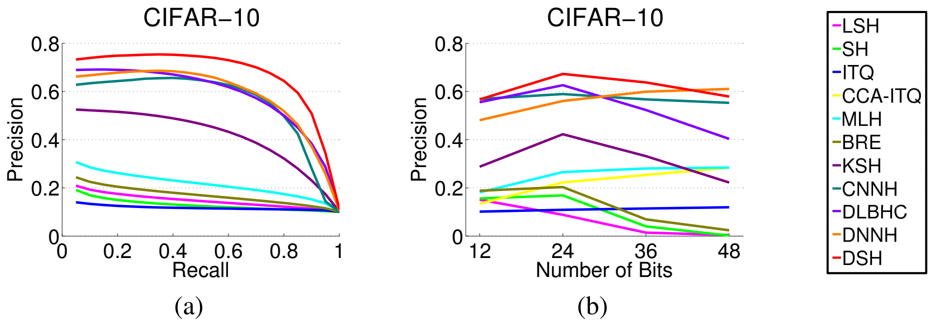

Figure 5. Comparison of retrieval performance of our DSH method and the other hashing methods on CIFAR-10 (results on NUS-WIDE are provided in supplementary materials). (a) PR curves (48-bit). (b) Mean precision within Hamming radius 2.

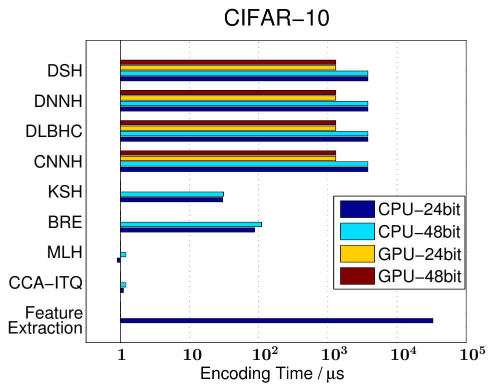

Figure 6. Time cost to encode one new-coming image (microseconds) on CIFAR-10.

The logarithmic encoding time (in microseconds, base 10) of such hashing methods is shown in Figure 6, where results were obtained by averaging over the whole test set. CNN-based methods take almost the same time to encode a single image with varying code lengths, since the computations mainly take place in the common preceding layers. In general, when only considering generating binary codes from model inputs, even the GPU accelerated version of CNN-based methods are slower than the conventional methods by at least an order of magnitude. However, taking the feature extraction time into consideration, the CNN-based methods are 10x faster than the conventional hashing methods. Moreover, the conventional hashing methods usually require several types of features to achieve comparable retrieval performance to CNN-based methods, which further slows down the whole encoding procedure.

5. Conclusion

We attribute the promising retrieval performance of DSH to three aspects: **First**, the coupling of non-linear feature learning and hash coding for extracting task-specific image representations; **Second**, the proposed regularizer for reducing the discrepancy between the real-valued network output space and the desired Hamming space; **Third**, the online generated dense pairwise supervision for well describing the desired Hamming space. In terms of efficiency, experiments have shown that the proposed method encodes new-coming images even faster than conventional hashing methods. Since our current framework is relatively general, more complex network structure can also be easily exploited. In addition, preliminary study of "network ensembles" in this work has proven it a promising way that is worth our future investigation to further boost retrieval performance.

Acknowledgements. This work is partially supported by 973 Program under contract No. 2015CB351802, Natural Science Foundation of China under contracts Nos. 61390511, 61379083, 61222211, 61272321, and Youth Innovation Promotion Association CAS No. 2015085.

References

[1] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, page 48, 2009. 5

[2] J. Deng, N. Ding, Y. Jia, A. Frome, K. Murphy, S. Bengio, Y. Li, H. Neven, and H. Adam. Large-scale object classification using label relation graphs. In *ECCV 2014*, pages 48–64. 2014. 1

[3] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *Computer Vision and Pattern Recognition (CVPR), 2015*, pages 2475–2483, 2015. 2, 5

[4] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999. 2, 7, 8

[5] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 249–256, 2010. 4

[6] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011*, pages 817–824, 2011. 2, 7, 8

[7] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer Vision and Pattern Recognition (CVPR), 2006*, pages 1735–1742, 2006. 3, 4

[8] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015. 1, 7

[9] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):117–128, 2011. 1

[10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678, 2014. 4

[11] A. Krizhevsky. Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, (4):7, 2009. 5

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 1, 3, 4, 7

[13] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems*, pages 1042–1050, 2009. 2, 7, 8

[14] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015*, pages 3270–3278, 2015. 2, 3, 4, 5, 7, 8

[15] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In *Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 27–35, 2015. 2, 3, 4, 7, 8

[16] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *Advances in Neural Information Processing Systems*, pages 3419–3427, 2014. 2, 3

[17] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012*, pages 2074–2081, 2012. 1, 2, 5, 7, 8

[18] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2015*, pages 3431–3440, 2015. 1

[19] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML-10*, pages 807–814, 2010. 4

[20] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML-11*, pages 353–360, 2011. 2, 7, 8

[21] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, 42(3):145–175, 2001. 5, 7

[22] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV 2012*, pages 876–889. 2012. 2

[23] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. *Computer Vision and Pattern Recognition (CVPR), 2015*, 2015. 2, 3

[24] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *Advances in Neural Information Processing Systems*, pages 1988–1996, 2014. 1

[25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR), 2015*, pages 1–9, 2015. 1, 7

[26] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In *Advances in Neural Information Processing Systems*, pages 2553–2561, 2013. 1

[27] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(12):2393–2406, 2012. 1, 2

[28] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, pages 1753–1760, 2008. 2, 7, 8

[29] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014. 2, 3, 4, 5, 7, 8

[30] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Transactions on Image Processing*, 24(12):4766–4779, 2015. 3, 4

[31] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2015*, pages 1556–1564, 2015. 2, 3, 4