DHA: Supervised Deep Learning to Hash with an Adaptive Loss Function

Jiehao Xu[1], Chengyu Guo[2] , Qingjie Liu[1*], Jie Qin[3], Yunhong Wang[1], Li Liu[2,4]
[1] State Key Laboratory of Virtual Reality Technology and System, Beihang University, China
[2] College of System Engineering, National University of Defense Technology, China
[3] Inception Institute of Artificial Intelligence
[4] Center for Machine Vision and Signal Analysis, University of Oulu, Finland

Abstract

Hashing, which refers to the binary embedding of high-dimensional data, has been an effective solution for fast nearest neighbor retrieval in large-scale databases due to its computational and storage efficiency. Recently, deep learning to hash has been attracting increasing attention since it has shown great potential in improving retrieval quality by leveraging the strengths of deep neural networks. In this paper, we consider the problem of supervised hashing and propose an effective model (i.e., DHA), which is able to generate compact and discriminative binary codes while preserving semantic similarities of original data with an adaptive loss function. The key idea is that we scale and shift the loss function to avoid the saturation of gradients during training, and simultaneously adjust the loss to adapt to different levels of similarities of data. We evaluate the proposed DHA on three widely-used benchmarks, i.e., NUS-WIDE, CIFAR-10, and MS COCO. The state-of-the-art image retrieval performance clearly shows the effectiveness of our method in learning discriminative hash codes for nearest neighbor retrieval.

1. Introduction

With the rapid development of the Internet and smart devices, a large volume of multimedia data such as videos and images are uploaded to the Internet every day. These data are with diverse contents and vary significantly in resolution, making it extremely difficult to find similar contents given users' request. This problem is known as (content-based) data retrieval. The term 'similar' may refer to visually or semantically similar. How to perform efficient retrieval in these high-dimensional data has become a problem worthy of attention. One solution to this is performing an approximate nearest-neighbor (ANN) search in the Hamming space by mapping high-dimensional visual data into compact binary codes with some elaborately designed hashing algorithms. Hashing has been receiving increasing attention from the community due to its efficiency in computation and storage of binary codes.

Recently, deep learning techniques have been proved to be effective in solving various vision tasks [11], inspiring many deep learning to hash methods [15, 10, 22, 24, 20, 27, 21, 9, 34, 33]. In this paper, we consider the problem of supervised hashing, which exploits similarity information as supervision to construct binary codes during learning, thus enabling better capturing the semantic structure of data. Most of existing supervised hashing methods leverage pairwise similarity tags to optimize the learning, which enforce the learned Hamming distances to align with the given tags by pushing the codes of similar data together and pulling the codes of dissimilar data away from each other. Normally, a sigmoid-alike loss function is employed to transform the Hamming distance to probability distribution that indicates their similarity. For dissimilar pairs, the Hamming distance between them is optimized to be greater than a threshold. However, a problem occurs when the distance surpasses a certain threshold, i.e., the gradient of their loss function is close to zero due to the saturation of the sigmoid function, making the training very difficult. In other words, this kind of loss function suffers from a high neighborhood ambiguity [2], which will degrade the retrieval performance.

In this paper, we argue that learning strategies of similar pairs and dissimilar pairs should be adaptive to the different similarities of data. As the similarity between data increases, the Hamming distance between their codes should be smaller. Furthermore, it is essential to minimize the Hamming distance between binary codes of very similar pairs. For dissimilar pairs, we only need to guarantee that the Hamming distance between them is greater than a certain threshold, rather than strictly maximizing it. With such an optimization scheme, the model will focus on learning the similarity of semantic information between similar data,

*Corresponding author is Qingjie Liu(qingjie.liu@buaa.edu.cn).

which is identical to the goal of searching and returning similar data in general retrieval tasks.

To deal with aforementioned problems, we propose a novel adaptive loss function that can scale and shift itself to avoid saturation of gradients during training. According to the similarity between data, the loss function impose varying degrees of constraints on distance between the hash codes reasonably, so that the hash model enables learning the Hamming distances more consistent with complex semantic similarity structures. Furthermore, we propose a new loss function to minimize the quantization loss, and introduce an improved cross-entropy loss to deal with class imbalance. Under the supervision of binary similarity tags or continuous similarity tags, our method achieves the state-of-the-art performance on three image databases, i.e., NUS-WIDE [6], CIFAR-10 [13], and MS COCO [19].

2. Related work

In the big data era, Hashing become a widely used approach for high-dimensional data retrieval, because of its excellent performance in approximate nearest neighbor search. Hashing methods can be generally divided into data-independent hashing and data-dependent hashing. Early studies on hashing mainly focused on data-independent methods, such as the locality sensitive hashing (LSH) family [8, 25], which randomly projects similar images to the same bucket with a high probability without using any training data. One major drawback of LSH is that long codes are required to attain a satisfactory search accuracy, which limits its application. To alleviate this issue, data-dependent hashing methods, which can produce more effective binary codes, emerge and significantly boost visual search performance. Because data-dependent hashing leverages machine learning tools and requires training data to parameterize the models, it is also known as learning to hash. The representative methods include spectral hashing [31], quantization hashing [10], etc. A recent study [1] has theoretically validated the performance advantage of data-dependent hashing. It can map high-dimensional data into Hamming space while preserving the original neighborhood structure, leading to improved retrieval quality. A variety of data-dependent hashing approaches have been proposed in recent years. In the following, we will give a brief review of unsupervised and supervised approaches of data-dependent hashing. For a comprehensive survey on learning to hash, please refer to [28, 29].

Unsupervised hashing methods learn hash functions by fitting the distribution of unlabeled training datasets. For example, Spectral Hashing (SH) [31] generates binary codes by minimizing the weighted Hamming distance of image pairs, where the weights are defined to be the similarity metrics of image pairs. Iterative Quantization(ITQ) [10] works on reducing the quantization loss during the binarization process.

In order to better maintain the semantic similarity in complicated high-dimensional data, supervised methods are proposed to take advantage of label information, such as similarity tags or category labels. Typical supervised methods include CCA-ITQ [10], which is an extension of ITQ, using supervision information to optimize the mapping of image descriptors. Supervised Hashing with Kernels (KSH) [20] proposes to learn similarity preserving hashing functions in kernel space by the pairwise relationship between examples. In deep Hashing, Convolutional Neural Networks Hashing (CNNH) [32] extremely maximizes the Hamming distances between binary codes of dissimilar images, and minimize the Hamming distances of similar images. HashNet [5] and DHN [36], ensure that the Hamming distance between hash codes of images is greater than a certain threshold. Later in [17], such a two-stage method with pairwise labels is further developed into an end-to-end system named Deep Pairwise-Supervised Hashing (DPSH), which performs simultaneous feature learning and hash encoding.

3. Approach

3.1. Problem Formulation

Suppose there is a training set with a size of N points, represented as: $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{N}$, and $\boldsymbol{x}_i$ is a D-dimensional feature vector $\boldsymbol{x}_i \in \mathbb{R}^D$. Each of which is associated with a binary code vector with K-dimension $\boldsymbol{b}_i \in \{-1, 1\}^K$, and $\boldsymbol{B} = \{\boldsymbol{b}_i\}_{i=1}^{N}$ is the set of binary vectors. Our goal of deep learning to hash is to learn a nonlinear hash mapping $\boldsymbol{\Phi} : \boldsymbol{X} \rightarrow \boldsymbol{B}$ from input space to Hamming space, such that the Hamming distance between hash codes of similar data will be small, and vice versa.

Usually, we can only obtain continuous representations. In other to obtain binary codes, the $\mathrm{sgn}(\cdot)$ function is constantly adopted to convert the continuous representations into binary codes $\boldsymbol{B}$. However, it will cause a fatal problem that we can not apply standard back-propagation method to learn the mapping $\boldsymbol{\Phi}$, since the $\mathrm{sgn}(\cdot)$ function is non-smooth, and it is indifferentiable at zero and zero gradient for all non-zero inputs, making it untrainable [12]. To relieve this problem, some approaches e.g. [5, 4, 30] approximate the $\mathrm{sgn}(\cdot)$ function with the smoothed $\tanh(\cdot)$. Following the similar relaxation strategy, instead of being optimized to generate discrete binary codes, we intend to learn continues approximation codes. Therefore, our goal becomes to learn a mapping $\boldsymbol{\Psi} : \boldsymbol{X} \rightarrow \boldsymbol{H}$, where $\boldsymbol{H} = \{\boldsymbol{h_i}\}_{i=1}^{N}, \boldsymbol{h_i} = (-1, 1)^K$, and $\boldsymbol{B} = \mathrm{sgn}(\boldsymbol{H})$. This leads to a relaxed optimization problem :

$$\{\boldsymbol{\Psi}, \boldsymbol{H}\} = \arg\min_{\boldsymbol{\Psi}, \boldsymbol{H}} (\mathcal{L}(\boldsymbol{H}) + \lambda \mathcal{Q}(\boldsymbol{H})) \tag{1}$$

where $\mathcal{L}(\boldsymbol{H})$ is the similarity loss, ensuring that the hash

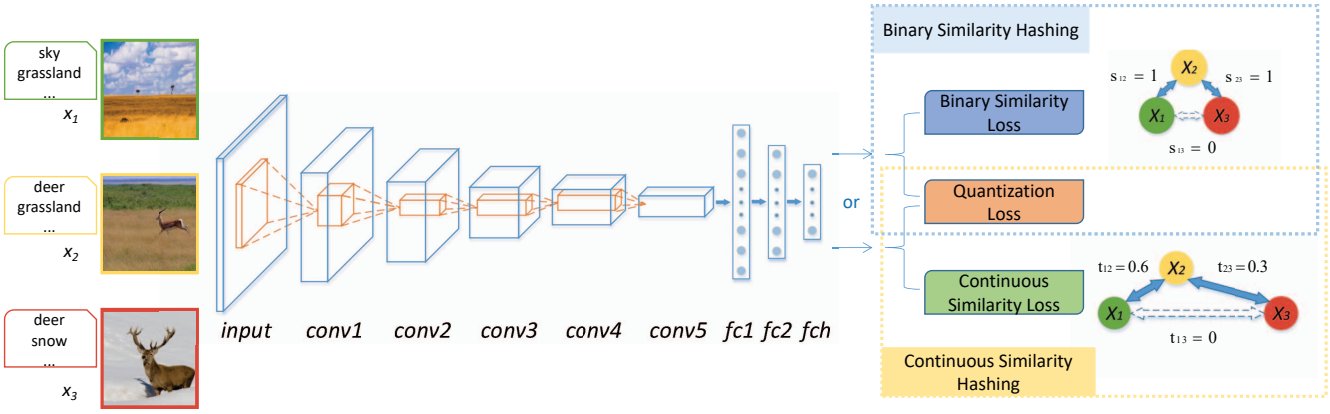

Figure 1. The architecture of proposed DHA. For binary similarity hashing, we use binary similarity tags $S = \{s\}$ as supervision, and adopt binary similarity loss and quantization loss to optimize our model. In this case, it may suffer from $d_H(\boldsymbol{x}_1, \boldsymbol{x}_3) \to 0$. For continuous similarity hashing, using continuous similarity tags $T = \{t\}$ and continuous similarity loss can learn more proper Hamming distance between binary codes.

codes can sufficiently preserve the semantic similarity of the data, and quantization loss $\mathcal{Q}(\boldsymbol{H})$ enforces that $\boldsymbol{H}$ approaches $\boldsymbol{B}$ to reduce the error caused by continuous relaxation. λ is a balance factor. In the following of this section, we will give detail description of these two losses.

3.2. Binary Similarity Hashing

3.2.1 Binary Similarity Loss

In this work, we focus on supervised deep learning to hash. To optimize the model, labels guiding the training process should be given. One solution is employing similarity tags of data pairs as supervision. Given two data points $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the similarity tag $s_{ij} \in \{0, 1\}$, in which $s_{ij} = 1$ if and only if $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ share at least one common semantic label, i.e. $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar, while $s_{ij} = 0$, $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are dissimilar. Based on this definition, we have a sample set $\{(\boldsymbol{x}_i, \boldsymbol{x}_j, s_{ij})\}$ for training, where each sample consists of a pair of data point and a similarity tag. This formulation is extensively used in previous studies [32, 36, 16].

For now, the goal of our learning to hash is to represent the input pair with K-bit compact codes $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$, while preserving their similarity information. To achieve this goal, supervised hashing is usually cast as a metric learning problem [23]: the model is optimized to minimize the Hamming distances of similar pairs and maximize the Hamming distances of dissimilar pairs. Since Hamming distance is indifferentiable, an alternative is taking inner product as substitution to compute Hamming distance [5, 35], which has the following form:

$$d_H(\boldsymbol{b}_i, \boldsymbol{b}_j) = \frac{1}{2}(K - d(\boldsymbol{b}_i, \boldsymbol{b}_j)) \tag{2}$$

where $d_H(\cdot, \cdot)$ represents the Hamming distance, and $d(\cdot, \cdot)$ is the inner product. This equation can be extended to the relaxation of $\boldsymbol{b}$, thus we have

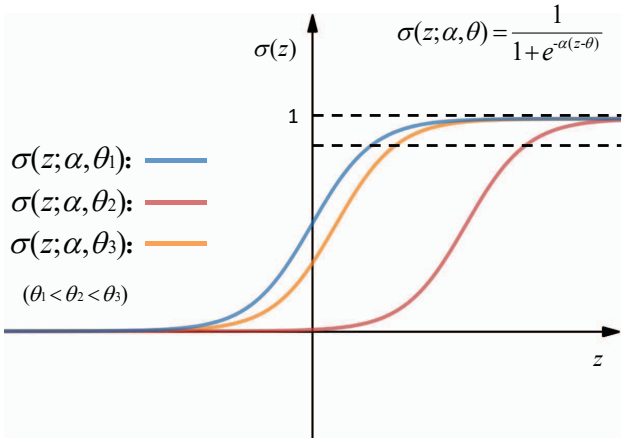

Figure 2. Function σ with different θ. With the increase of θ, it enforces larger z to get a equally high value of $\sigma(z)$.

$$d_H(\boldsymbol{h}_i, \boldsymbol{h}_j) = \frac{1}{2}(K - d(\boldsymbol{h}_i, \boldsymbol{h}_j)) \tag{3}$$

Throughout this paper, we adopt this equation to compute distances of codes. If data pair $(\boldsymbol{x}_i, \boldsymbol{x}_j)$ are similar, the inner product of $\boldsymbol{h}_i$ and $\boldsymbol{h}_j$ (and $d(\boldsymbol{b}_i, \boldsymbol{b}_j)$) should be large (approach K).

Binary Similarity. Given a pair of data $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ and their codes $\boldsymbol{h}_i$, $\boldsymbol{h}_j$, to estimate similarity s_{ij} of them, we can define the following likelihood function

$$p(s_{ij}|\boldsymbol{h}_i, \boldsymbol{h}_j) = \begin{cases} \sigma(d(\boldsymbol{h}_i, \boldsymbol{h}_j)), & \text{if } s_{ij} = 1 \\ 1 - \sigma(d(\boldsymbol{h}_i, \boldsymbol{h}_j)), & \text{if } s_{ij} = 0 \end{cases} \tag{4}$$

where $\sigma(\cdot)$ is a function scaling the inner product into a distribution, normally it takes sigmoid function form defined as $\sigma(z) = \frac{1}{1+e^{-z}}$. In this paper we introduce a new generalization of σ:

$$\sigma(z) = \frac{1}{1 + e^{-\alpha(z-\theta)}} \tag{5}$$

As $d(\cdot, \cdot)$ increase, $p(s_{ij} = 1)$ will be larger and approach 1, giving us a high confidence to determine that the two examples $\boldsymbol{x}_i$, $\boldsymbol{x}_j$ are similar, vice verse. Note that there are two hyper-parameters α and θ. α is a hyper-parameter introduced in [5] to control bandwidth of sigmoid function. Greater α gives rise to larger saturation zone where its gradient is zero, causing severe gradient vanishing. As suggested in [5], α should be smaller than 1. We set it as $\frac{10}{K}$. Recall that, for similar pairs, we expect their Hamming distance as zero as possible, equivalently the inner product of them should be large enough to be close to K. Although we have α to relieve gradient vanishing, it will inevitably reach saturation zone of sigmoid function when K is far from zero. This problem will become serious when dealing with very large scale data retrieval, because longer coding bits is required to ensure a better performance. To conquer this challenge, we add a shift parameter θ into the sigmoid function to push the saturation zone away from the range of $d(\boldsymbol{h}_i, \boldsymbol{h}_j)$. To be specific, if $s_{ij} = 1$, we set $\theta > 0$, as θ_3 in Fig. 2, because it requires a large $d(\boldsymbol{h}_i, \boldsymbol{h}_j)$ to obtain high probability $p(s_{ij} = 1)$. Correspondingly, we set θ equals to 0, if $s_{ij} = 0$, because we expect a small $d(\boldsymbol{h}_i, \boldsymbol{h}_j)$ and it will fall out of the saturation zone. Comparing with previous methods, we impose different constraints on $s_{ij} = 1$ and $s_{ij} = 0$, resulting in smaller Hamming distance between codes of similar examples and relatively larger Hamming distance between codes of dissimilar examples.

Hard Examples. Given Eqn. 4, we employ the binary Cross Entropy (CE) as the loss function to optimize the proposed model. A problem occurs when training the network, which is there exist hard examples that cannot be fixed well because the summation of vast majority easy examples will overwhelm those hard ones, leading to ineffective training for hard examples. Inspired by [18], we adopt Focal Loss to make learning focus on hard examples by introducing modulating factor $(1-p)^2$

$$\begin{aligned} FL(p) &= -(1-p)^2 \log(p) \\ &= \begin{cases} -(1-\sigma)^2 \log(\sigma), & \text{if } s_{ij} = 1 \\ -\sigma^2 \log(1-\sigma), & \text{if } s_{ij} = 0 \end{cases} \end{aligned} \tag{6}$$

Intuitively, the modulating factor reduces the loss contribution from easy examples and extends the range in which an example receives low loss. When an example is misclassified and p is small, the modulating factor is near 1 and the loss is unaffected as CE. As $p \to 1$, the factor goes to 0 and the loss for well-classified examples is down-weighted.

Data Imbalance. Since the training samples are constructed with the semantic labels of data, a huge number of dissimilar pairs can be generated, which is far more than similar pairs, resulting in extreme data imbalance. To solve this problem, we set a weight factor:

$$\omega_{ij} = \begin{cases} \beta, & \text{if } s_{ij} = 1 \\ 1 - \beta, & \text{if } s_{ij} = 0 \end{cases} \tag{7}$$

where $\beta \in (0, 1)$ and greater than 0.5. ω_{ij} will increase the weight of similar examples.

Besides the advantage of alleviating data imbalance problem, another benefit of inserting ω_{ij} into the loss function is that it enables a better retrieval performance. Because it drives the network to focus on learning the similarity of pairs, which is consistent with the practical application that we want to return more similar raw data in retrieval problem.

By combining the Eqn. 4, 6 and 7, we finally obtain our adaptive loss with respect to binary similaritry:

$$\begin{aligned} \mathcal{L}(\boldsymbol{H}) = \sum_{s_{ij} \in S} -\omega_{ij} [& s_{ij}(1-\sigma)^2 \log(\sigma) \\ & + (1 - s_{ij})\sigma^2 \log(1-\sigma)] \end{aligned} \tag{8}$$

where S is the set of binary similarity tags.

3.2.2 Quantization Loss

We apply the $\tanh(\cdot)$ function to squash the output of the network to be within $(-1, 1)$ and then obtain the compact codes $\boldsymbol{h}$. To reduce quantization errors between continues $\boldsymbol{h}$ and discrete $\boldsymbol{b}$, we introduce a novel quantization loss $\mathcal{Q}(\boldsymbol{H})$,

$$\mathcal{Q}(\boldsymbol{H}) = \sum_{i=1}^{N} \sum_{j=1}^{K} \frac{1}{K} (1 - e^{(|\boldsymbol{h}_i^j| - 1)}) \tag{9}$$

where $\boldsymbol{h}_i^j$ is the jth element of vector $\boldsymbol{h}_i$, $|\boldsymbol{h}_i^j|$ represents the absolute value of $\boldsymbol{h}_i^j$. The closer $|\boldsymbol{h}_i^j|$ is to 1, the smaller the loss function will be. By optimizing the $\mathcal{Q}(\boldsymbol{H})$ function during training, the value of $\boldsymbol{h}_i^j$ will approach to {-1,1}, which helps to reduce the quantization error caused by $\boldsymbol{B} = \text{sgn}(\boldsymbol{H})$.

3.3. Continuous Similarity Hashing

Binary similarity tag $s \in \{0, 1\}$ can only model simple semantic similarity. When data are rich in content and labeled with multiple semantic labels, representing similarity relationship for them may leads to ambiguity results. Suppose there are three samples $\boldsymbol{x}_i$, $\boldsymbol{x}_j$, and $\boldsymbol{x}_k$, all with multiple semantic labels. $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, $\boldsymbol{x}_j$ and $\boldsymbol{x}_k$ share at least one common semantic labels thus they are similar, with $s_{ij} = 1$,

$s_{jk} = 1$. While $\boldsymbol{x}_i$ and $\boldsymbol{x}_k$ do not share share any label, they are not similar, thus $s_{ik} = 0$. Since we optimize the model to minimize the Hamming distance of similar pairs as zero as possible: $d_H(\boldsymbol{b}_i, \boldsymbol{b}_j) \to 0$ and $d_H(\boldsymbol{b}_j, \boldsymbol{b}_k) \to 0$, this may result in $d_H(\boldsymbol{b}_i, \boldsymbol{b}_k) \to 0$. The model makes a wrong decision $s_{ik} = 1$, which is contrary to the fact that $s_{ik} = 0$, as shown in Fig. 1. Single label based similarity do not have this issue, since if two of them are similar then all of them must be similar. To deal with this problem induced by multiple semantic labels, we propose to introduce continuous similarity tags t to account for the continuous nature of latent semantic space.

3.3.1 Construction of Continuous Similarity Tags

We construct continuous similarity tags of pairs from their semantic labels. Let $\boldsymbol{y}_i \in \{0, 1\}^C$ denotes label vector for an data sample $\boldsymbol{x}_i$, and C represents the number of categories. The continuous similarity tag of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ can be constructed as

$$t_{ij} = \cos(\boldsymbol{y}_i, \boldsymbol{y}_j) = \frac{\boldsymbol{y}_i \cdot \boldsymbol{y}_j}{||\boldsymbol{y}_i|| \times ||\boldsymbol{y}_j||} \tag{10}$$

When $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are not assigned with common label attributes, it means $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are dissimilar, and t_{ij} equals to 0. On the contrary, t_{ij} will approach 1, when $\boldsymbol{x}_i$ is highly similar to $\boldsymbol{x}_j$.

3.3.2 Continuous Similarity Loss

Comparing with binary similarity tags $s \in \{0, 1\}$, $t \in [0, 1]$ can represent more complex similarities between data samples. Furthermore, by taking advantage of t, it is possible to better preserve the fine-grained features of similar data samples. To achieve this end, we control the value of θ in function $\sigma = 1/(1 + e^{-\alpha(z-\theta)})$ according to tags t:

(1) In the case of $t_{ij} > 0$, to minimize $\mathcal{L}(\boldsymbol{H})$, $p(s_{ij} = 1) = \sigma(d(\boldsymbol{h}_i, \boldsymbol{h}_j))$ is supposed to be maximized.

> (**a**) When $t_{ij} \to 1$, we apply a large value of θ, as θ_3 in Fig. 2. In order to achieve a high confidence p, the value of $d(\boldsymbol{h}_i, \boldsymbol{h}_j)$ should approach K, contributing to $d_H(\boldsymbol{b}_i, \boldsymbol{b}_j) \to 0$.
> (**b**) When $t_{ij} \to 0$, a small value of θ, such as θ_2 in Fig. 2 is adopted in σ. It is relatively easy for p to obtain a high confidence value, which means $d_H(\boldsymbol{b}_i, \boldsymbol{b}_j) \to 0$ is no longer strictly required, a relatively small value of which will give high probability to determine they are similar.

(2) As for $t_{ij} = 0$, the objective is minimizing $\sigma(d(\boldsymbol{h}_i, \boldsymbol{h}_j))$ and a much smaller θ, such as θ_1 will be adopted in σ. It enforces smaller $d(\boldsymbol{h}_i, \boldsymbol{h}_j)$ to get a low value of σ, resulting in larger $d_H(\boldsymbol{b}_i, \boldsymbol{b}_j)$. In summary, by imposing different degrees of constraint on data pairs of various similarity, our approach is able to adaptively learn the optimal Hamming distances between data pairs.

4. Experiments

In this section, the experiment results are presented to validate the performance of our method in image retrieval.

4.1. Experimental Setup

We conducted extensive experiments on three widely used image retrieval datasets: NUS-WIDE [6], MS COCO [19] and CIFAR-10 [13].

NUS-WIDE is a public Web image dataset which contains 269,648 images collected from Flickr. Each of these images is manually annotated with one or multiple labels in 81 concepts (categories) for evaluating retrieval models. We randomly sample 5,000 images to construct a test set, and the remaining images used as the database. Furthermore, 10,000 images from the database are sampled as the training set.

MS COCO is a large-scale object detection, segmentation, and captioning dataset. It contains 82,783 training images and 40,504 validation images where each image is associated with some labels of 80 categories. Following [5], images with no category information are pruned, and this leaves us 12,2218 images for training and validation. 5,000 images are sampled as queries, with the rest of the images used as the database. 10,000 images from the database made up the training set.

CIFAR-10 consists of 60,000 images in 10 classes, with 6,000 images per class. There are 50,000 images in the training set and 10,000 test images in the test set. Each image is single-labeled by one of the 10 categories. We follow protocol in [3] to randomly select 100 images per class as query set, 500 images per class as training set, and the remaining images are used as the database. Two images are similar if they belong to the same class.

As mentioned above, the similarity tags for hash learning and evaluation is constructed from image labels. (1) Binary similarity tags s: if two images $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are assigned with at least one common label, they are similar and s_{ij} = 1; otherwise, they are dissimilar and s_{ij} = 0. (2) Continuous similarity tags t: by calculating the cosine similarity of label vectors $(\boldsymbol{y}_i, \boldsymbol{y}_j)$, we obtain a continuous tag indicating the similarity degree of two images $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. It should be noted that labels are not indispensable, as long as similarity information is available, our DHA method can learn effective hash codes.

For evaluation metric, we use variants of standard mean Average Precision (mAP). For binary similarity hash, we compare DHA against both classical and recent state-of-the-art hashing methods. These methods include: unsupervised methods LSH [8], SH [31], ITQ [10], supervised shallow

Table 1. Binary similarity experiments on multi-label datasets, NUS-WIDE, and MS COCO. Our proposed method achieves state-of-the-art performances in all bits of hash codes.

mAP@5000	NUS-WIDE				MS COCO			
Method	16bit	32bit	48bit	64bit	16bit	32bit	48bit	64bit
LSH [8]	0.328	0.423	0.433	0.501	0.459	0.486	0.544	0.585
BRE [15]	0.503	0.529	0.548	0.555	0.592	0.622	0.630	0.634
ITQ [10]	0.509	0.543	0.558	0.561	0.582	0.624	0.646	0.657
ITQ-CCA [10]	0.460	0.405	0.373	0.347	0.566	0.562	0.530	0.509
SDH [26]	0.476	0.555	0.579	0.581	0.555	0.564	0.572	0.580
CNNH [32]	0.570	0.583	0.593	0.600	0.564	0.574	0.571	0.567
DNNH [16]	0.598	0.616	0.635	0.639	0.593	0.603	0.605	0.610
DNH [36]	0.637	0.664	0.669	0.671	0.677	0.701	0.695	0.694
HashNet[5]	0.663	0.699	0.711	0.716	0.687	0.718	0.730	0.736
DHA	**0.669**	**0.706**	**0.721**	**0.727**	**0.708**	**0.731**	**0.741**	**0.752**

Table 2. Binary similarity experiments on single-label dataset, CIFAR-10.

mAP@54000	CIFAR-10			
Method	16bit	32bit	48bit	64bit
BRE [15]	0.370	0.438	0.438	0.491
ITQ-CCA [10]	0.354	0.414	0.449	0.462
KSH [20]	0.524	0.559	0.567	0.569
SDH [26]	0.461	0.520	0.553	0.568
CNNH [32]	0.476	0.472	0.489	0.501
DNNH [16]	0.559	0.558	0.581	0.583
DNH [36]	0.568	0.603	0.621	0.635
HashNet [5]	0.643	0.667	0.675	0.687
DHA	**0.652**	**0.681**	**0.690**	**0.699**

methods BRE [15], KSH [20], ITQ-CCA [10], SDH [26], and supervised deep learning to hash methods CNNH [32], DNNH [16], DHN [36] and HashNet [5]. For continuous similarity hash, we mainly compare DHA with HashNet-C, which is a variant of HashNet that uses continuous similarity tags.

We implement our method based on the PyTorch platform, which is one of the widely adopted deep learning frameworks. For fair comparisons, we adopt the AlexNet [14] as the backbone for feature extraction. We take the outputs of $fc7$ layer from AlexNet as image representation, and replace the final softmax classification layer with a new fully-connected hash layer fch to produce hash codes. We use the mini-batch stochastic gradient descent algorithm to optimize the network, where the batch size is set to 256, the weight decay is 0.0005 and momentum is 0.9. We initialize the first five convolutional layers and two fully connected layers with weights pre-trained on ImageNet [7], and the newly added fch layer with random numbers drawn from a zero-centered Gaussian distribution. We fine-tune the backbone network on our task with a start learning rate of 0.001, and decrease it with a factor of 0.1 as training proceed. The learning rate of fch layer is set to be 10 times greater than the backbone network. Because the ratio between the number of dissimilar pairs and the number of similar pairs is roughly 10, 5, and 1 for CIFAR-10, NUS-WIDE, and MS COCO, we set β as $\frac{11}{12}$, $\frac{6}{7}$, $\frac{2}{3}$, respectively, to deal with the data imbalance.

4.2. Results

We evaluate retrieval quality based on standard evaluation metrics mean Average Precision (mAP), and we adopt mAP@5K(mAP evaluated on the top 5,000 retrievals) for NUS-WIDE and MS COCO and mAP@54,000 for CIFAR-10, respectively.

4.2.1 Binary Similarity Experiments

Table 1 presents the results of hashing methods on multi-label datasets, NUS-WIDE, and MS COCO. In most cases, supervised hashing methods such as SDH and CNNH outperform unsupervised hashing methods. Unsupervised method ITQ also performs well in experiments. Moreover, deep learning-based supervised hashing methods such as HashNet, DNH, and DNNH surpass most non-deep hashing methods. The primary reason may be that deep hashing method are able to extract better image features, and they are also able to learn feature representation and the hashing jointly.

As illustrated in Table 1, our proposed method DHA,

Table 3. Continuous similarity experiments on multi-label dataset, NUS-WIDE and MS COCO. HashNet+C is a variant using continuous similarity tags. DHA-t is our proposed method that use continuous similarity tags as supervision. Both two method have significant improvement.

mAP@5000	NUS-WIDE				MS COCO			
Method	16bit	32bit	48bit	64bit	16bit	32bit	48bit	64bit
HashNet	0.663	0.699	0.711	0.716	0.687	0.718	0.730	0.736
HashNet+C	0.665	0.702	0.721	0.726	0.688	0.726	0.737	0.742
DHA	0.669	0.706	0.721	0.727	0.708	0.731	0.741	0.752
DHA+t	**0.683**	**0.716**	**0.728**	**0.733**	**0.711**	**0.742**	**0.756**	**0.768**

Table 4. Ablation Study on NUS-WIDE and MS COCO.

mAP@5000	NUS-WIDE				MS COCO			
Method	16bit	32bit	48bit	64bit	16bit	32bit	48bit	64bit
DHA	0.669	0.706	0.721	0.727	0.708	0.731	0.741	0.752
DHA-W	0.643	0.677	0.691	0.708	0.703	0.728	0.737	0.749
DHA-Q	0.601	0.644	0.653	0.676	0.668	0.677	0.684	0.706

Figure 3. Examples of top 10 retrieved results in MS COCO. Retrieved images marked with green boxes are correct, while the rest of the images marked with red boxes are incorrect.

outperforms all of competing methods on multi-label datasets and achieves great performances in all bits of hash codes. For example, comparing with the highly competitive HashNet, DHA surpasses it by $0.5\% - 2\%$ on both datasets. The improvement of DHA is much more significant in terms of long hash codes on MS COCO. The improvement indicates that imposing different constraints on the Hamming distances between binary codes based on the similarity of images improves the performance in multi-label datasets.

Retrieval results for CIFAR-10 are shown in Table 2. CIFAR-10 is a single-label dataset with fewer images, and it only contains 10 class, we retrieve all the 54,000 images in the database to compute the mAP. The proposed method DHA surpasses the nearest competitor HashNet in terms of all hash bits.

As discussed above, our method can generate compact binary hash codes and achieves state-of-the-art performances on all three databases. By training with the proposed adaptive loss function, we can better preserve the semantic similarity of images.

4.2.2 Continuous Similarity Experiments

Since the continuous similarity tags $\{t\}$ are constructed from multi-label of images and CIFAR-10 is a single-label dataset, we only report our results on two multi-label databases (i.e., NUS-WIDE and MS COCO), which are given in Table 3. Multi-label is more informative, and it is essential to exploit this information when performing practical retrieval task in the real application. We compare our approach with HashNet+C [5], a variant of HashNet that uses continuous similarity tags similar to ours. DHA-t is

our method trained with continuous similarity tags t. As illustrated in Table 3, DHA-t surpasses HashNet+C on both datasets, and the improvement of DHA-t is much more significant on MS COCO, especially retrieval with more bits. Compared with the original DHA, it is obvious that using continuous similarity labels as supervision can generate more effective binary codes capturing complex similarity structures between images.

4.2.3 Ablation Study

We conduct ablation study in this section by dropping balance factor ω and quantization loss function $\mathcal{Q}$: (1) DHA-W is a variant without balance factor ω , meaning that $\beta = 0.5$; (2) DHA-Q, is a variant that does not use quantization loss to reduce quantization errors.

From the table4, we can notice that in the absence of the balance factor ω, our method has shown a decline on both databases. In particular, the performance of DHA-W on the NUS-WIDE declines drastically. The ratio between the number of dissimilar pairs and the number of similar pairs for NUS-WIDE is 5, which is highly imbalanced. On MS COCO, the ratio is about 1; thus, the two results are comparable. There is only a little drop for DHA-W on this dataset. Without quantization loss, the mAP of DHA-Q on both databases drop significantly. By reviewing outputs of DHA-Q, we find out that some of the bits in the outputs are close to zero, resulting in unstable quantization when using $\mathrm{sgn}(\cdot)$ function. Therefore, reducing the quantization error when binarizing continuous representations to binary codes is crucial for hashing methods.

5. Conclusion

We have proposed a novel supervised hashing method with an adaptive loss function for approximate nearest neighbor retrieval. By imposing different constraints on the Hamming distances between binary codes based on the semantic similarities, our model could be trained steadily and reduce the ambiguity of neighborhood in constructing the Hamming space. Experiments have shown that DHA could generate compact binary codes and yield the state-of-the-art retrieval performance on three datasets, i.e., NUS-WIDE, CIFAR-10, and MS COCO.

6. Acknowledgments

This work was supported by the Foundation for Innovative Research Groups through the National Natural Science Foundation of China under Grant 61421003.

References

[1] A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 793–801. ACM, 2015. 2

[2] F. Cakir, K. He, S. A. Bargal, and S. Sclaroff. Hashing with mutual information. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 1

[3] Y. Cao, B. Liu, M. Long, and J. Wang. Hashgan: Deep learning to hash with pair conditional wasserstein gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1287–1296, 2018. 5

[4] Y. Cao, M. Long, B. Liu, and J. Wang. Deep cauchy hashing for hamming space retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1229–1237, 2018. 2

[5] Z. Cao, M. Long, J. Wang, and P. S. Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5608–5617, 2017. 2, 3, 4, 5, 6, 7

[6] 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. ACM, 2009. 2, 5

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 6

[8] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999. 2, 5, 6

[9] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 484–491, 2013. 1

[10] 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*, 35(12):2916–2929, 2012. 1, 2, 5, 6

[11] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew. Deep learning for visual understanding: A review. *Neurocomputing*, 187:27–48, 2016. 1

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 2

[13] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 2, 5

[14] 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. 6

[15] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in neural information processing systems*, pages 1042–1050, 2009. 1, 6

[16] H. Lai, P. Yan, L. Ye, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. 2015. 3, 6

[17] W.-J. Li, S. Wang, and W.-C. Kang. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855*, 2015. 2

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017. 4

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2, 5

[20] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2074–2081. IEEE, 2012. 1, 2, 6

[21] X. Liu, J. He, B. Lang, and S.-F. Chang. Hash bit selection: a unified solution for selection problems in hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1570–1577, 2013. 1

[22] M. Norouzi and D. M. Blei. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th international conference on machine learning*, pages 353–360. Citeseer, 2011. 1

[23] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov. Hamming distance metric learning. In *Advances in neural information processing systems*, pages 1061–1069, 2012. 3

[24] M. Norouzi, A. Punjani, and D. J. Fleet. Fast search in hamming space with multi-index hashing. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3108–3115. IEEE, 2012. 1

[25] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in neural information processing systems*, pages 1509–1517, 2009. 2

[26] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 37–45, 2015. 6

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

[28] J. Wang, W. Liu, S. Kumar, and S.-F. Chang. Learning to hash for indexing big data—a survey. *Proceedings of the IEEE*, 104(1):34–57, 2015. 2

[29] J. Wang, T. Zhang, N. Sebe, H. T. Shen, et al. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):769–790, 2017. 2

[30] X. Wang, Y. Shi, and K. M. Kitani. Deep supervised hashing with triplet labels. In *Asian conference on computer vision*, pages 70–84. Springer, 2016. 2

[31] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009. 2, 5

[32] 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, 6

[33] F. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *International conference on machine learning*, pages 946–954, 2014. 1

[34] P. Zhang, W. Zhang, W.-J. Li, and M. Guo. Supervised hashing with latent factor models. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 173–182. ACM, 2014. 1

[35] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:1556–1564, 2015. 3

[36] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016. 2, 3, 6