10-2019

Deep hashing by discriminating hard examples

Cheng YAN

Guansong PANG
Singapore Management University, gspang@smu.edu.sg

Xiao BAI

Chunhua SHEN

Jun ZHOU

See next page for additional authors

Citation

YAN, Cheng; PANG, Guansong; BAI, Xiao; SHEN, Chunhua; ZHOU, Jun; and HANCOCK, Edwin. Deep hashing by discriminating hard examples. (2019). *Proceedings of the 27th ACM International Conference on Multimedia, Nice, France, 2019 October 21-25*. 1535-1542.
Available at: https://ink.library.smu.edu.sg/sis_research/7139

Author
Cheng YAN, Guansong PANG, Xiao BAI, Chunhua SHEN, Jun ZHOU, and Edwin HANCOCK

Deep Hashing by Discriminating Hard Examples

Cheng Yan[1] Guansong Pang[2] Xiao Bai[1] Chunhua Shen[2] Jun Zhou[3] Edwin Hancock[4,1]
[1]School of Computer Science and Engineering, Beijing Advanced Innovation Center for Big Data and Brain Computing, Beihang University
[2]School of Computer Science, University of Adelaide
[3]School of Information and Communication Technology, Griffith University
[4]Department of Computer Science, University of York
email: [1]{beihangyc, xiaobai}@buaa.edu.cn, [2]{guansong.pang, chunhua.shen}@adelaide.edu.au,
[3]jun.zhou@griffith.edu.au, [4]edwin.hancock@york.ac.uk

Abstract

This paper tackles a rarely explored but critical problem within learning to hash, i.e., to learn hash codes that effectively discriminate hard similar and dissimilar examples, to empower large-scale image retrieval. *Hard similar examples* refer to image pairs from the same semantic class that demonstrate some shared appearance but have different fine-grained appearance. *Hard dissimilar examples* are image pairs that come from different semantic classes but exhibit similar appearance. These hard examples generally have a small distance due to the shared appearance. Therefore, effective encoding of the hard examples can well discriminate the relevant images within a small Hamming distance, enabling more accurate retrieval in the top-ranked returned images. However, most existing hashing methods cannot capture this key information as their optimization is dominated by *easy examples*, i.e., distant similar/dissimilar pairs that share no or limited appearance. To address this problem, we introduce a novel Gamma distribution-enabled and symmetric Kullback-Leibler divergence-based loss, which is dubbed *dual hinge loss* because it works similarly as imposing two smoothed hinge losses on the respective similar and dissimilar pairs. Specifically, the loss enforces exponentially variant penalization on the hard similar (dissimilar) examples to emphasize and learn their fine-grained difference. It meanwhile imposes a bounding penalization on easy similar (dissimilar) examples to prevent the dominance of the easy examples in the optimization while preserving the high-level similarity (dissimilarity). This enables our model to well encode the key information carried by both easy and hard examples. Extensive empirical results on three widely-used image retrieval datasets show that (i) our method consistently and substantially outperforms state-of-the-art competing methods using hash codes of the same length and (ii) our method can use significantly (e.g., 50%-75%) shorter hash codes to perform substantially better than, or comparably well to, the competing methods.

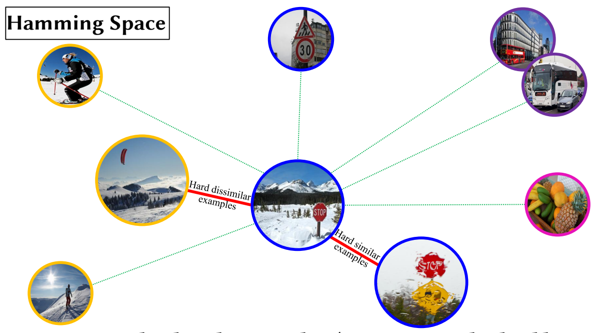


Figure 1: Exemplar hard examples (image pairs linked by red lines) in the COCO data. The hard dissimilar example is the image pairs that come from different semantic classes, person and stop sign, but exhibit similar appearance. The hard similar example is the image pairs from the same semantic class, stop sign, which demonstrate some shared appearance but have different fine-grained appearance. By contrast, easy examples are distant image pairs (linked by green lines) that share no or limited appearance.

Keywords

Image Retrieval; Deep Hashing; Hard Examples; Hinge Loss

ACM Reference Format:
Cheng Yang, Guansong Pang, Xiao Bai, Chunhua Shen, Jun Zhou, and Edwin Hancock. 2019. Deep Hashing by Discriminating Hard Examples. In *Proceedings of the 27th ACM Internationnal Conference on Multimedia (MM'19), Oct. 21-25, 2019, Nice, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3343031.3350927

1 Introduction

Nearest neighbor (NN) search has been widely adopted in computer vision and machine learning. However, the time complexity of NN methods on a dataset of size n is $O(n)$, making the exact NN search infeasible in real-world large-scale retrieval applications. To address this problem, approximate nearest neighbor (ANN) techniques have been proposed. The idea is to locate approximate nearest neighbors rather than the exact neighbors. Due to its low storage cost and fast retrieval speed, hashing has been widely used as a solution to ANN problems [2, 3, 10, 11, 23, 28, 30–34, 36, 37, 41, 42].

The goal of hashing is to construct hash functions to map each data point from the original data space into a Hamming space with associated binary codes so that visually similar samples are mapped into similar binary codes (as measured in terms of Hamming distance). By using the binary code-based data representations, both the storage cost and time complexity for search are dramatically reduced. In fact, the NN search can be achieved within a constant or sub-linear time complexity. Numerous hashing methods have been introduced [8, 13, 23–25, 27, 36–39, 43]. Among all these methods, deep learning to hash methods [1, 5, 6, 8, 22, 24, 25, 27, 43] have

This work was done when Cheng Yan was visiting University of Adelaide.
Xiao Bai is the corresponding author.

achieved remarkable success in image retrieval and gained substantially better performance than traditional hashing methods due to their end-to-end learning frameworks, in which convolutional neural networks-based feature learning and hash coding functions can be well unified to encode any nonlinear hash function.

Most of these existing methods [5, 8, 24, 25, 27, 43] focus on learning hash codes using the high-level pairwise similar/dissimilar information, without considering the importance of different image pairs. However, some image pairs inherently convey much more critical information than the others. For example, learning hash codes to effectively distinguish a dissimilar image pair with one image containing a cat and another containing a tiger generally facilitates a more discriminative power than a dissimilar pair of images containing a cat and an elephant respectively. Therefore, substantially more expressive hash codes can be learned by well leveraging these important image pairs. However, on the other hand, since those important image pairs have only fine-grained difference, they are often within a small Hamming distance, which makes them very difficult to be properly learned and exploited in the current popular pairwise-based hashing techniques.

To address this issue, we introduce a rarely explored but critical problem within learning to hash, i.e., to learn hash codes that effectively discriminate hard similar and dissimilar examples, to empower large-scale image retrieval. *Hard similar examples* refer to image pairs from the same semantic class that demonstrate some shared appearance but have different fine-grained appearance. *Hard dissimilar examples* are image pairs that come from different semantic classes but exhibit similar appearance. Accordingly, *easy examples* are distant similar/dissimilar pairs that share no or limited appearance. Some exemplar hard and easy examples are given in Figure 1. The hardness is originally due to the difficulty to effectively encode the fine-grained appearance in these close image pairs. Therefore, effective discrimination of the hard examples enables accurate ANN search of truly relevant images within a small Hamming distance.

We further tackle this problem with a novel deep hashing method, which explicitly discriminates the hard examples to learn their fine-grained difference, in addition to preserving the similarity or dissimilarity of easy examples. This objective is achieved by a novel Gamma distribution-enabled and symmetric Kullback-Leibler divergence-based loss. The loss is dubbed *dual hinge loss* (DHL for short) because it works similarly as imposing two smoothed hinge losses on the respective similar and dissimilar pairs. The first analogous hinge loss exerts on similar pairs: it enforces exponentially variant penalization on the hard similar examples and equally large bounding penalization on easy similar examples. This enables the model to emphasize and learn the fine-grained difference of hard similar examples while also preserving the high-level similarity information. The second analogous hinge loss exerts on dissimilar pairs: it exponentially increases the penalization with decreasing distance of the hard dissimilar examples and prunes the loss of easy dissimilar examples. This helps preventing the dominance of the easy examples in the optimization while also preserving the high-level dissimilarity.

An early exploration of a similar problem was done in [5], which learns hash codes to concentrate on similar images to be within a small Hamming distance. It achieves this goal by imposing significantly large penalization on similar image pairs with Hamming distance larger than a given Hamming radius threshold. This is similar to enforcing large penalization on the easy examples so as to highlight the importance of the hard similar examples in our case. However, their penalization on the easy examples increases rapidly with the increasing Hamming distance. As a result, the optimization may be still dominated by the easy examples and the hard examples are often ignored during training. By contrast, our loss function can well discriminate the hard examples while at the same time effectively prevent the dominance of the easy examples by bounding penalization on both similar and dissimilar easy examples.

In summary, this paper makes the following major contributions.

- We introduce a rarely explored but critical problem within learning to hash, i.e., to learn hash codes that can effectively discriminate hard examples while preserving the proximity information of easy examples.
- We further propose a novel loss, called dual hinge loss (DHL), to tackle this problem. This loss enables our deep hashing model to not only preserve the similarity or dissimilarity of the distant image pairs (i.e., easy examples) but also effectively learn the fine-grained difference of the close image pairs (i.e., hard examples).
- We also present a novel Gamma quantization loss function to minimize the loss of converting prediction probabilities into hash codes.

Comprehensive experiments on three widely used image retrieval datasets, NUS-WIDE, MS-COCO and CIFAR-10, show that (i) the proposed method outperforms the best performer among eight state-of-the-art competing methods by a significant margin of 3.8%-8.7% and (ii) our method is able to use significantly (e.g., 50%-75%) shorter hash codes to perform substantially better than, or comparably well to, the competing methods.

1.1 Related Work

Existing supervised hash methods [5, 8, 15, 25, 27, 29, 35] take full advantage of the semantic information, e.g. pairwise similarity or relevance feedback, which normally achieve better accuracy than unsupervised methods. Recently, deep learning based hashing methods [5, 8, 15, 24, 25, 27, 43, 44] have shown superior performance over traditional hashing methods in which joint learning methods have been proposed [24, 43] to unify the two steps to learn more lossless encoding.

To further improve performance, cross-entropy [5, 8, 17, 25] has been adopted to preserve the pairwise similarity of the data in the Hamming space. To this end, different types of loss functions have been proposed [4, 7, 8, 17, 43]. However, the resulting hash codes often fail to discriminate the hard and easy examples. This is because they enforce significant penalization on the easy examples with a very large Hamming distance and treat the hard examples and the other easy examples equally. As a result, they are ineffective in properly encoding the image pairs within small Hamming distance. Cauchy distribution is incorporated into the loss function in [5] to address this problem, but its penalization on the easy examples increases quickly with increasing Hamming distance. As a result, the optimization is often dominated by easy examples with large Hamming distance, leading to less discriminative binary encoding.

Additionally, some previous methods [4, 5, 8] attempt to explicitly assign different weights to image pairs to highlight important examples. These methods treat image pairs from small classes as more important examples than that from the large classes, and incorporate the class proportion as sample weights into their loss function. Unlike these methods that consider the sample importance from the class imbalance aspect, we address a fundamentally different problem, which defines the example importance based on the hardness of correctly distinguishing the image pairs.

2 The Proposed Method

2.1 Problem Statement

In a retrieval system, $\mathcal{X} = \{x_i\}_{i=1}^{N}$ be a set of N training samples, $\mathcal{S}$ be the pairwise similarity label set for M pairs of samples x_i and x_j, where for each $s_{ij} \in \mathcal{S}$ we have $s_{ij} = 1$ if x_i and x_j are similar pairs rand $s_{ij} = 0$ otherwise, then a general deep learning to hash problem is to learn a nonlinear hash function $\phi : \mathcal{X} \mapsto \mathcal{H}$ which maps the original data points $\mathcal{X}$ to a Hamming space $\mathcal{H}$ of the hash codes with k bit hash codes, i.e., $\mathcal{H} \in \{-1, 1\}^k$. The hash function $h_i = \phi(x_i)$ is designed to preserve the proximity information of $\mathcal{X}$ in the Hamming space $\mathcal{H}$ with $\mathcal{S}$ as the supervision information. Given a query sample x_q, the retrieval system first obtains its hashing representation $h_q = \phi(x_q)$ and then returns a set of samples that are close to $h_q = \phi(x_q)$ in $\mathcal{H}$. Thus, faithful hashing is the key to the success of an ANN-based retrieval system. Particularly, the similarity between x_q and x_i is evaluated by the Hamming distance using their hash codes h_q and h_i, so it only requires $O(1)$ time to search all neighbors within a certain Hamming radius by hash table lookup instead of a linear scan. One main issue here is that the problem is too general. Many detailed knowledge embedded in $\mathcal{X}$ may therefore be ignored, leading to ineffective retrieval performance.

We introduce a critical subproblem within the above general hashing problem to address this issue, which aims to effectively learn the fine-grained difference of hard examples while also preserving the high-level proximity information. Specifically, the set of hard examples can be formally defined as $\mathcal{D}$, where $(x_{d_i}, x_{d_j}) \in \mathcal{D}$ is a close similar/dissimilar pair within a small Hamming distance r; while the set of easy examples is accordingly defined as $\mathcal{E}$, where $(x_{e_i}, x_{e_j}) \in \mathcal{E}$ is a distant similar/dissimilar pair with a Hamming distance larger than r. Then, the new problem is to learn a hash function $\phi : \mathcal{D} \cup \mathcal{E} \mapsto \mathcal{H}$ in a way that ϕ puts more focus on $\mathcal{D}$ while also effectively encodes the information carried by $\mathcal{E}$. This problem is critical for a retrieval system, because effective encoding of fine-grained difference is prominent for the system to achieve its ultimate goal, i.e., to return the truly relevant results from a set of highly similar results w.r.t. a given query. However, this problem is specially challenging because (i) $\mathcal{D}$, $\mathcal{E}$ and r are

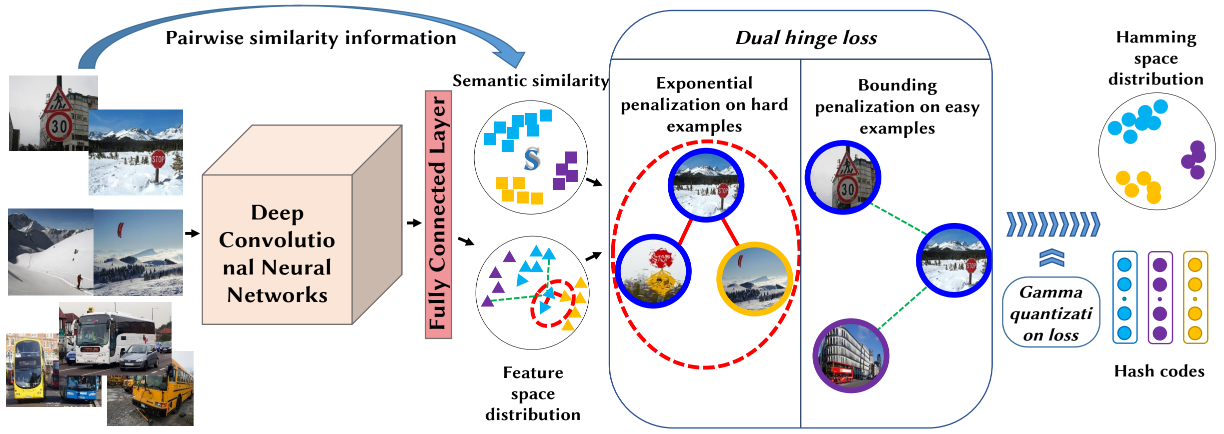

Pairwise similarity information
Semantic similarity
Deep Convolutional Neural Networks
Fully Connected Layer
Feature space distribution
Dual hinge loss
Exponential penalization on hard examples
Bounding penalization on easy examples
Hamming space distribution
Gamma quantization loss
Hash codes

Figure 2: An overview of the proposed deep hashing method. It is an end-to-end deep hashing approach with three modules, including deep convolutional network-based feature learning, dual hinge loss-based hashing learning and Gamma quantization loss. The feature learning starts with a deep convolutional neural network, followed by a k-dimensional fully connected layer and a hyperbolic tangent function. The second module uses the proposed dual hinge loss to learn the fine-grained difference of hard examples and high-level proximity of easy examples in Hamming space. The last module defines a Gamma quantization loss to reduce the quantization error.

often unknown in practice and (ii) it is very difficult to define different loss functions to effectively learn the information in both $\mathcal{D}$ and $\mathcal{E}$.

This paper introduces a novel loss called dual hinge loss to well address the two main challenges of this new problem. This loss performs analogously to a dual hinge loss to incorporate the key information in both $\mathcal{D}$ and $\mathcal{E}$ into our learned hash codes. Also, the parameter r is automatically determined by the hyper-parameters of the loss, facilitating an easy and effective way of setting r.

2.2 Our Framework

The proposed method is an end-to-end hashing learning framework that aims to well discriminate the hard examples in $\mathcal{D}$ for a more lossless mapping from raw image data into the Hamming space. As shown in Figure 2, the method consists of three main modules, including deep convolutional network-based feature learning, dual hinge loss-based hashing learning and Gamma quantization loss. Specifically, it first uses a convolutional neural network for feature learning of each inputs, which can be any deep structure with convolutional neural layers such as AlexNet [20] or ResNet [16], followed by a k-dimensional fully-connected layer. The hyperbolic tangent function is used in the fully-connected layer to control the scale of feature within the interval $[-1, 1]$, which helps reduce the gap between real-value feature and hash codes. Our method then leverages the proposed dual hinge loss to learn the fine-grained difference of hard examples and high-level proximity of easy examples in Hamming space. Lastly, it defines a novel Gamma quantization loss to reduce the quantization error for further assurance of the quality of hash codes. Below we introduce the last two modules.

2.3 Dual Hinge Loss

The dual hinge loss is a synthesis of a novel symmetric KL divergence-based loss and a Gamma distribution-based distance-to-probability function. The loss enforces exponential penalization on hard examples and bounding penalization on easy examples, which enables us to learn both the fine-grained difference of the hard examples and the high-level proximity information carried by easy examples.

2.3.1 Symmetric KL Divergence We introduce a symmetric KL divergence based loss to maintain the distribution of data points by jointly preserving the similarity of pairwise data points for deep hashing. The general Kullback-Leibler(KL) divergence is defined as $\sum_i p_i \log \frac{p_i}{q_i}$, which is equivalent to the cross-entropy minus the entropy. A larger KL divergence indicates a greater difference between two different distributions. However, KL divergence is inherently asymmetric, so it is commonly used in an extended version for hashing learning [14, 18, 40], defined as:

$$\begin{aligned} KL(P\|Q) &= \theta_e KL(P\|Q) + (1-\theta_e)KL(Q\|P) \\ &= \sum_{(x_i,x_j)\in\mathcal{S}} \left[\theta_e p_{ij}\log\frac{p_{ij}}{q_{ij}} + (1-\theta_e)q_{ij}\log\frac{q_{ij}}{p_{ij}}\right], \end{aligned} \tag{1}$$

where θ_e is a parameter to balance the two terms. In general, $\theta_e = 0.5$ is used because it becomes a distance metric only when $\theta_e = 0.5$, and $KL(P\|Q) \neq KL(Q\|P)$ otherwise. Furthermore, KL does not have a finite upper bound, which makes it difficult to accurately measure the difference between distributions.

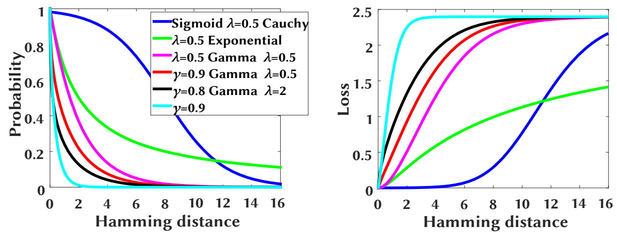
To address this issue, we propose a novel symmetric KL divergence based loss, namely SKL, which is defined as follows:

$$\begin{aligned} SKL(P\|Q) = &\sum_{(x_i,x_j)\in\mathcal{S}} p_{ij}\log\frac{\theta p_{ij}}{(\theta-1)p_{ij}+q_{ij}} \\ &+ \sum_{(x_i,x_j)\in\mathcal{S}} q_{ij}\log\frac{\theta q_{ij}}{(\theta-1)q_{ij}+p_{ij}}, \end{aligned} \tag{2}$$

where $\theta \geq 1$ is a parameter to control the scale of $SKL(P\|Q)$. Two desired properties of SKL are that: (i) it is a symmetric metric since we always have $SKL(P\|Q) = SKL(Q\|P)$ and (ii) it has a θ-dependent upper bound. Additionally, SKL is a generalized loss of the symmetric KL loss in Eqn. (1) with $\theta_e = 0.5$, since it is a specific form of SKL with $\theta = 1$.

Suppose that the pairwise semantic relationship $\mathcal{S}$ follows a distribution Q while the corresponding hash code-based relationship follows a distribution P, then SKL can be used to learn the hash codes. Specifically, the distribution Q is fixed since the supervised information $\mathcal{S}$ is known *a priori*, so our aim is to learn the distribution P and force it to be as similar as possible as Q by Eqn. (2). For each p, if p is close to 1, the distance between the hash codes of two data points is small in Hamming space; and if p is close to 0, the distance is large. To well control the sensitivity to the distance, we adopt the well-known Gamma distribution as a probability function φ of p, which is elaborated in the next subsection.

2.3.2 Gamma Probability-enabled symmetric KL-based Loss The distributions of distances between hash codes needs to be transformed into a probability distribution for subsequent encoding. In principle any probability function φ can be used to instantiate our model. Previous state-of-the-art deep learning to hash methods usually adopt an adaptive Sigmoid function $\varphi = \frac{1}{1+e^{-\lambda x}}$ [4, 8, 43] which is controlled by parameter λ. However, the probability of the Sigmoid function is still too high when the Hamming distance becomes large, as shown in Figure 3. In [5] this issue is handled by using the Cauchy distribution $\frac{1/\lambda}{1/\lambda+x}$, which helps put more focus on the hard examples. However, its penalization on the easy examples is not bounded, and its loss on the hard examples is still not sufficiently sensitive.



(a) Probability (b) Loss

Figure 3: The value of (a) probability and (b) our SKL loss w.r.t. Hamming distance between the hash codes of similar pairs. One desired property of the Gamma distribution based probability is that its probability is exponentially sensitive to changes within a small Hamming radius and become insensitive to large Hamming distance. We shift each line in (b) so they start with the same point for better visibility. Accordingly, our Gamma distribution-enabled SKL loss significantly penalizes similar pairs within small Hamming distance and flattens out for large Hamming distance.

To enable greater sensitivity to distance, we propose a novel probability function based on Gamma distribution:

$$p_{ij} = c\, d(h_i, h_j)^{\gamma-1} e^{-\lambda d(h_i, h_j)}, \tag{3}$$

where γ and λ are hyper-parameters, $d(h_i, h_j)$ is the distance between the hash codes h_i and h_j, and c is a constant. p_{ij} in Eqn. (3) is monotone decreasing when $d(h_i, h_j) > 0$ and $\gamma <= 1$. In practice, we add a tiny constant c_0 to $d(h_i, h_j)$, i.e., $d(h_i, h_j) = d(h_i, h_j) + c_0$, in order to satisfy the condition of $d(h_i, h_j) = 0$, and set $c = c_0^{1-\gamma} e^{\lambda c}$ so that the function can well complete the distance-to-probability transformation. In Eqn. (3), the larger the Hamming distance, the smaller the conditional probability p_{ij}, and vice versa. Its sensitivity to distance can be controlled by parameters λ and γ.

As shown in Figure 3, our Gamma probability function significantly penalizes similar image pairs within a small Hamming distance i.e. hard similar examples. Particularly, the probability/loss is more sensitive at the beginning, since the value of Eqn. (3) changes exponentially, leading to greater sensitivity than both of the Sigmoid function and the Cauchy function. The exponential function is a special case of our Gamma function with $\gamma = 1$ and we can change its parameters to obtain different rate of change.

Combining Eqns. (2) and (3), we obtain the dual hinge loss as:

$$\begin{aligned} L_{dhl} = &\sum_{(x_i,x_j)\in\mathcal{S}} c d_{ij}^{\gamma-1} e^{-\lambda d_{ij}} \log \frac{\theta c d_{ij}^{\gamma-1} e^{-\lambda d_{ij}}}{(\theta-1) c d_{ij}^{\gamma-1} e^{-\lambda d_{ij}} + s_{ij}} \\ &+ \sum_{(x_i,x_j)\in\mathcal{S}} s_{ij} \log \frac{\theta s_{ij}}{(\theta-1) s_{ij} + c d_{ij}^{\gamma-1} e^{-\lambda d_{ij}}}. \end{aligned} \tag{4}$$

Below we have some theoretical and empirical analysis of the key properties of this loss.

Theorem 2.1 (Key Properties of Dual Hinge Loss). *In L_{dhl}, the loss for dissimilar pairs monotonically and exponentially decreases with increasing Hamming distance, with a bounding loss of 0. By contrast, the loss for similar pairs monotonically and exponentially increases with increasing Hamming distance, with a bounding loss of* $\log \frac{\theta}{(\theta-1)}$*, where θ is a hyper-parameter of the SKL loss in Eqn. (2).*

Proof. Since we have $q \in \{0, 1\}$ in Eqn. (2) based on the pairwise similarity information, Eqn. (2) becomes $p \log \frac{\theta}{(\theta-1)}$ when $q = 0$, which is a monotonically and exponentially increasing function. When $q = 1$, Eqn. (2) becomes $p \log \frac{\theta p}{(\theta-1)p+1} + \log \frac{\theta}{(\theta-1)+p}$, which is a monotonically and exponentially decreasing function.

When the Hamming distance of each pair increases, p decreases to 0. According to Eqn. (3), for $q = 0$ we obtain a lower bound for $p \log \frac{\theta}{(\theta-1)}$ as

$$\lim_{p\to 0} p \log \frac{\theta}{(\theta-1)} = 0. \tag{5}$$

Accordingly, for $q = 1$ we obtain an upper bound of the loss as

$$\lim_{p\to 0} p \log \frac{\theta p}{(\theta-1)p+1} + \log \frac{\theta}{(\theta-1)+p} = \log \frac{\theta}{(\theta-1)}. \tag{6}$$

□

This theorem states that our loss essentially imposes a loss function which is analogous to a exponentially smoothed hinge loss to the dissimilar pairs, and applies a loss function which is analogous to an *inverse* exponentially smoothed hinge loss to the similar pairs. Such a dual smoothed hinge loss effectively bounds the penalization on easy examples while significantly penalizes hard examples. Moreover, this loss well unifies the determination of hard and easy examples and its hyper-parameters, so they can be easily tuned via cross-validation.

As illustrated in Figure 4, for dissimilar pairs on the left panel, our loss function bounds the loss for easy and well-distinguished examples (e.g., with Hamming distance greater than 5) and puts more focus on hard dissimilar examples having small distance, e.g., Hamming distance smaller than 5. This way enables the model to faithfully preserve the original dissimilarity while preventing the dominance of the easy examples in the optimization. For similar pairs on the right panel, our loss function enforces equally large penalization on distant similar image pairs and significantly punishes close similar image pairs to learn their fine-grained difference. Compared with previous methods [4, 5, 7, 8, 17, 43], our dual hinge loss is more effective for discriminating the hard examples while also preserving the proximity of the easy examples.

2.4 Relaxation and Gamma Quantization Loss

Since optimizing the sign function is difficult due to its ill-posed gradient, it is necessary to replace the Hamming distance with an appropriate approximation on continuous codes. In our method, we adopt the normalized Euclidean distance to approximate the Hamming distance for a pair of binary hash codes h_i and h_j, which is defined as:

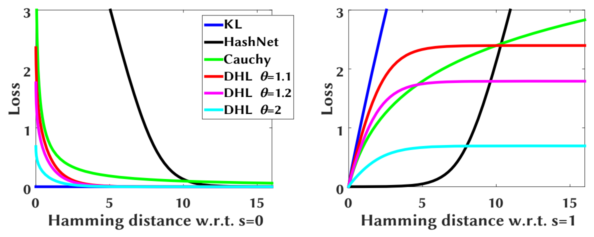
$$\begin{aligned} d(h_i, h_j) &= \frac{k}{4} \left\| \frac{f_i}{\|f_i\|} - \frac{f_j}{\|f_j\|} \right\|_2^2 \\ &= \frac{k}{2}(1 - \cos(f_i, f_j)). \end{aligned} \tag{7}$$

where k is the code length, f_i is the feature of x_i from the fully-connected layer, $h_i = sign(f_i)$ and $\cos(f_i, f_j) = \frac{f_i \cdot f_j}{\|f_i\|\|f_j\|}$ is the cosine of two vectors f_i and f_j. We can see that Eqn. (7) can preserve the angle of two data points to a certain extent after relaxation.

We further introduce a Gamma quantization loss as follows:

$$L_Q = \sum_{i=1}^{N} \frac{e^{\lambda d(|h_i|, 1)}}{d(|h_i|, 1)^{\gamma-1}}, \tag{8}$$

where $d(|h_i|, 1)$ can be taken to be either the Hamming distance between the hash codes or the Euclidean distance between the continuous codes. L_Q is used to achieve a lossless quantization, and so we can use continuous values for the hash codes during training. For testing, we can directly obtain the binary hash codes by using sign function $sign(h_i)$ to obtain k-bit binary codes for each sample.



(a) Dissimilar pairs (b) Similar pairs

Figure 4: The value of the loss w.r.t. Hamming distance for (a) dissimilar pairs and (b) similar pairs. On the left, with parameter $\theta > 0$, our loss function DHL reduces the loss for distant image pairs with Hamming distance larger than 5 (i.e., easy examples) to be close to zero, and puts more focus on hard dissimilar examples. Similarly, on the right panel, it enforces equally large penalization on easy similar examples and exponentially variant penalization on hard similar examples to learn the fined-grained difference. By contrast, current state-of-the-art losses, such as HashNet [8] and Cauchy [5], do not have such properties. Note that KL is ill-posed when $s = 0$.

We then obtain our final loss function:

$$L = L_{dhl} + \alpha L_Q \tag{9}$$

where L_{dhl} and L_Q are from Eq. 4 and Eq. 8, α is a parameter to balance the two losses.

3 Experiments

3.1 Datasets and Evaluation Measures

Our Dual Hinge Loss based Hashing (**DHLH**) was evaluated on three widely-used publicly available datasets, including NUS-WIDE [9], MS-COCO [26], and CIFAR-10 [19].

NUS-WIDE contains 269, 648 images from Flickr with each image having some of the 81 ground truth concepts. Following [5, 8], we randomly sampled 5, 000 images as the query samples, with the remaining images used as the database. We randomly sampled 10, 000 images from the database as the training samples.

MS-COCO contains 82, 783 training images and 40, 504 validation images, where each image is labeled with some of the 80 semantic concepts. Following the protocol in [8], we combined the training and validation sets to obtain a collection of 123, 287 images. We randomly sampled 5, 000 images as the query samples. The rest images were used as the database, in which 10, 000 images were sampled for training.

CIFAR-10 contains 60, 000 images in 10 classes. Following the protocol in [5, 43], we randomly selected 100 images per class as the query set and 500 images per class as the training set, with the rest of other images as the database for retrieval.

Following the standard protocol in [5, 8, 43], the pairwise similarity information is constructed from image labels for hash learning and performance evaluation. If two images x_i and x_j share at least one common label, they are a similar pair, i.e., $s_{ij} = 1$; and they are a dissimilar pair otherwise, with $s_{ij} = 0$.

To have a comprehensive performance evaluation, four different evaluation metrics were used, including Mean Average Precision (MAP), Precision-Recall (PR) curves, Precision curves within Hamming distance 2 ($P@Hr \leq 2$), and Precision curves w.r.t. the number of top returned samples ($P@N$). For fair comparison, all methods used exactly the same training and test sets.

3.2 Competing Methods and Their Settings

The retrieval performance of DHLH is compared with both traditional shallow and recently emerging deep learning-based methods. Since the performance of all shallow methods such as ITQ-CCA [12], BRE [21], KSH [29] and SDH [35] are significantly less effective than deep learning-based methods, we only report the results of SDH and KSH which have relatively better performance than the other two methods. Six state-of-the-art deep hashing methods are chosen, including DNNH [22], DHN [43], DPSH [24], HashNet [8], DCH [5], and HashGAN [4].

The shallow hashing methods were built upon feature representations extracted using AlexNet [20]. All deep learning based hashing methods take raw images as inputs and use the same network architecture, AlexNet with last layer replaced by a k-dimensional fully-connected layer using a hyperbolic tangent function as their feature representation learner.

In terms of optimization, the AlexNet model was pre-trained on ImageNet, and then our method DHLH trained the last layer and fine-tuned all the layers through back-propagation. Since the last layer was trained from scratch, its learning rate was set to be 10 times larger than that of the lower layers. The mini-batch size was set to 128. We used mini-batch stochastic gradient descent with 0.9 momentum and the weight decay parameter set to 10^{-4}. The learning rate was cross-validated from 10^{-5} to 10^{-1} with a multiplicative step $\sqrt{10}$. The competing deep methods were optimized in the same way or as recommended as their authors. The model parameters of DHLH and the parameters of each method under comparison were selected by cross-validation.

3.3 Results

3.3.1 Overall Retrieval Performance The MAP results for our DHLH method and the eight competing methods on the three datasets are reported in Table 1. We evaluated all methods with four different lengths of hash codes. DHLH consistently and substantially outperforms all the eight competing methods across four different code lengths. Particularly, DHLH achieves significant improvements over the best performance of the competing methods by 6.2%-8.7% (16 bits), 3.8%-5.1% (32 bits), 4.2%-4.6% (48 bits) and 4.3%-4.8% (64 bits). Also, DHLH can use significantly shorter hash codes (i.e., 16 or 32 bits) to perform substantially better than, or comparably well to, the best competing methods such as HashNet, DCH and HashGAN using 64-bit hash codes. The superior performance of our model is mainly due to its strong capability to effectively encode the key information carried by both easy and hard examples. As expected, the performance of the deep learning-based methods is much better than the traditional methods.

3.3.2 Performance of Hamming Space Retrieval Since the Hamming space retrieval only requires O(1) time for each query, the performance in terms of Precision within Hamming radius 2 ($P@Hr = 2$) is very important for efficient retrieval with binary hash codes. The $P@Hr = 2$ performance w.r.t. different code lengths of all the methods is shown in Figure 5. DHLH is clearly the best performer on all three datasets. In particular, the $P@Hr = 2$ performance of DHLH with 16 bits is much better than the competing methods. This reinforces the importance of DHLH to leverage the hard examples to learn the fine-grained difference of relevant images within a small Hamming distance, which is the main driving force of learning more discriminative hash codes. Most hashing methods achieve the best accuracy with moderate code lengths. This is because when using longer codes, the Hamming space would become sparse and few data points fall within the Hamming ball with radius 2.

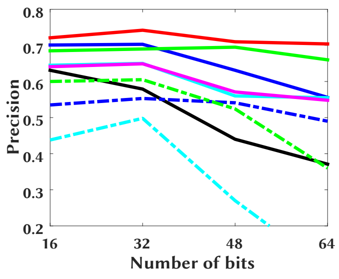
3.3.3 Top-N Precision Performance The retrieval performance in terms of Precision-Recall curves (PR) and Precision curves w.r.t. different numbers of top returned samples ($P@N$) on three datasets are shown in Figure 6 and Figure 7 respectively. It can be seen that our DHLH method outperforms all the competing methods. Impressively, compared to the competing methods, DHLH achieves much higher precision at low recall, as shown in Figure 6. More importantly, as shown in Figure 7, DHLH performs substantially better in the top-ranked returned results . This is a desired capability for many precision-first retrieval systems. Note that the retrieval task on CIFAR-10 is relatively easier than on the other two datasets, so the top performers DHLH, HashGAN and DCH have very comparable performance in the PR curves in Figure 6.

3.4 Analysis of the Proposed Method

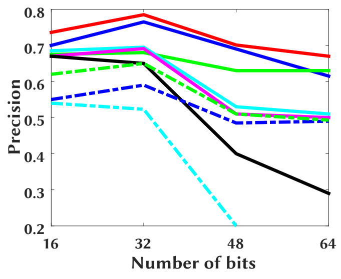
3.4.1 Ablation Study DHLH is compared with its five variants to examine the contribution of the two key components of dual hinge loss and the network architecture. (i) DHLH-O is a DHLH variant replacing the symmetric KL loss with original KL divergence loss; (ii) DHLH-M is a DHLH variant replacing the symmetric KL loss with maximum a posteriori [5, 8]; (iii) DHLH-S is a DHLH variant replacing the Gamma distribution based probability with a adaptive Sigmoid function used in [8, 24]; (iv) DHLH-C is

Table 1: MAP of different methods on image retrieval task. The best performance per column is boldfaced.

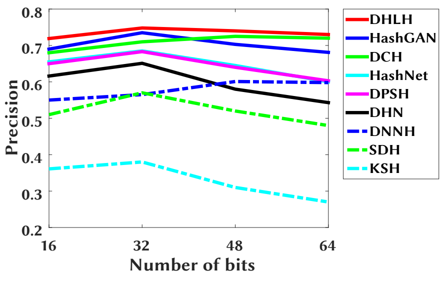
Methods	NUS-WIDE				MS-COCO				CIFAR-10			
	16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits
KSH	0.550	0.580	0.613	0.635	0.520	0.533	0.534	0.535	0.521	0.553	0.560	0.565
SDH	0.587	0.612	0.633	0.670	0.556	0.563	0.574	0.579	0.460	0.518	0.551	0.569
DNNH	0.598	0.617	0.633	0.639	0.587	0.596	0.603	0.611	0.556	0.560	0.576	0.583
DHN	0.635	0.666	0.670	0.671	0.678	0.683	0.688	0.691	0.572	0.598	0.621	0.633
DPSH	0.601	0.616	0.623	0.680	0.699	0.713	0.728	0.740	0.713	0.726	0.743	0.751
HashNet	0.662	0.698	0.711	0.716	0.685	0.712	0.728	0.734	0.643	0.667	0.673	0.685
DCH	0.712	0.735	0.739	0.743	0.703	0.716	0.733	0.740	0.681	0.730	0.734	0.742
HashGAN	0.714	0.738	0.744	0.749	0.697	0.725	0.741	0.744	0.667	0.731	0.737	0.749
DHLH	**0.761**	**0.767**	**0.778**	**0.785**	**0.745**	**0.762**	**0.773**	**0.780**	**0.725**	**0.759**	**0.768**	**0.781**



(a) NUS-WIDE

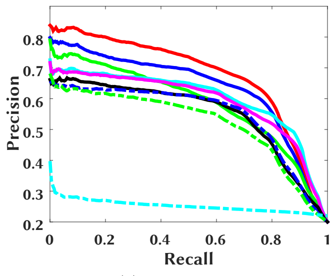


(b) MS-COCO

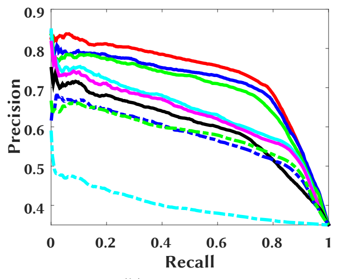


(c) CIFAR-10

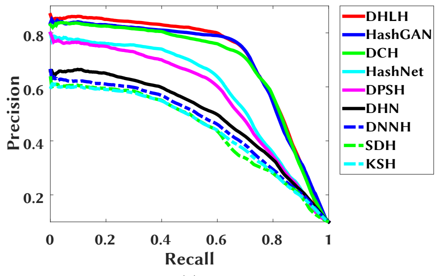
Figure 5: The Precision curves within Hamming Radius 2 of DHLH and competing methods on the three benchmark datasets.



(a) NUS-WIDE

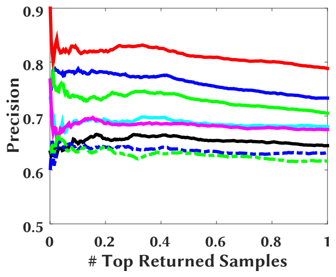


(b) MS-COCO

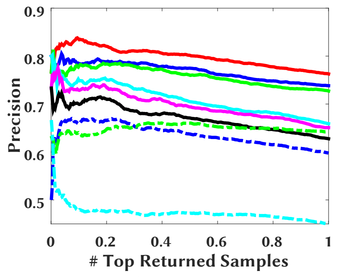


(c) CIFAR-10

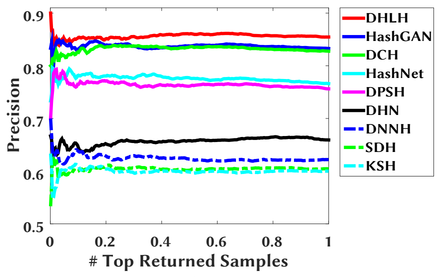
Figure 6: The Precision-Recall curve @64 bits of DHLH and competing methods on the three benchmark datasets.



(a) NUS-WIDE



(b) MS-COCO



(c) CIFAR-10

Figure 7: Precision curve w.r.t. top-N @64 bits of DHLH and competing methods on the three benchmark datasets.

a DHLH variant replacing the Gamma distribution based probability with a Cauchy function [5]; (v) DHLH-R is a DHLH variant replacing the AlexNet structure with ResNet-50 [16].

Importance of SKL loss. The MAP results w.r.t. different code lengths on the three datasets are reported in Table 2. DHLH achieves respectively 11.4%-15.8% and 2.1%-4.1% average improvements over DHLH-O and DHLH-M on the three datasets using four different code lengths. This demonstrates

Table 2: MAP of different DHLH variants on image retrieval task. The best performance per column is boldfaced.

Methods	NUS-WIDE				MS-COCO				CIFAR-10			
	16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits
DHLH	0.761	0.767	0.778	0.785	0.745	0.762	0.773	0.780	0.725	0.759	0.768	0.781
DHLH-O	0.655	0.663	0.672	0.680	0.668	0.683	0.696	0.701	0.633	0.648	0.665	0.686
DHLH-M	0.724	0.741	0.748	0.756	0.724	0.737	0.751	0.760	0.705	0.743	0.755	0.770
DHLH-S	0.713	0.739	0.749	0.753	0.705	0.725	0.743	0.754	0.700	0.725	0.745	0.756
DHLH-C	0.725	0.748	0.762	0.766	0.709	0.741	0.756	0.765	0.707	0.738	0.751	0.759
DHLH-R	**0.790**	**0.801**	**0.808**	**0.811**	**0.773**	**0.776**	**0.785**	**0.796**	**0.779**	**0.796**	**0.807**	**0.813**

the effectiveness of using our SKL loss to enforce a bounding penalization on easy examples to prevent the dominance of the easy examples in the optimization.

Importance of Gamma distribution-based probability. On average, DHLH also substantially outperforms DHLH-S by a large margin of respective 6.5%, 5.7% and 4.7% on NUS-WIDE, MS-COCO and CIFAR-10, and outperforms DHLH-C with respective 5.0%, 5.1% and 2.9%. This underlines the large improvement of using the Gamma distribution based probability function. The improvement is due to its exponential sensitivity to the pairwise Hamming distance within a small Hamming radius.

Feature learning architecture. In terms of feature learning, we find that replacing AlexNet with ResNet-50 results in remarkable improvements on all three datasets. This demonstrates the capability of DHLH in enhancing its performance by using more advanced network architectures.

3.4.2 Parameter Sensitivity Analysis We examined the sensitivity of four hyper-parameters θ, α, γ and λ, which were respectively set to 1.1, 0.001, 0.9, 1 by default. We varied one of them with the other three fixed for each experiment. θ was searched in the range of $[1.001, 2]$. Similarly, α was examined in $[10^{-5}, 10^{-1}]$, γ was examined in $[0.01, 1]$, and λ was examined in $[0.05, 2]$. The results are shown in Figure 8.

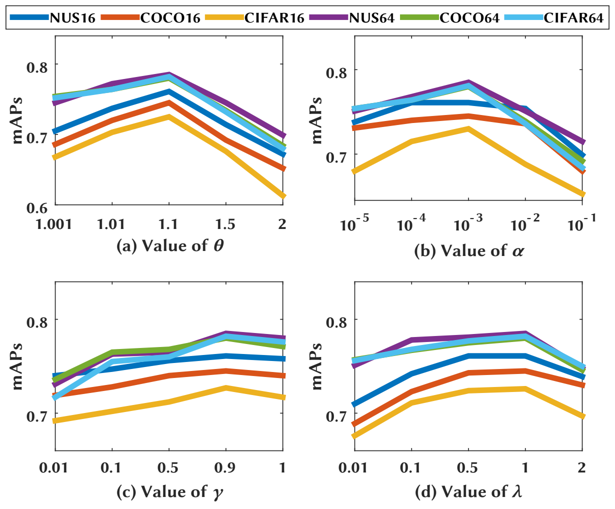


Figure 8: The MAP results w.r.t. θ, α, γ and λ using 16 and 64 bits hash codes on three datasets.

As shown in Figure 8 (a), the MAP performance of DHLH first increases steadily and then drops quickly. This is because the parameter θ determines the upper bounding loss of easy similar examples. Small θ leads to insufficient penalization on the easy examples, while large θ leads to too large penalization on these easy examples, and as a result, the easy examples dominate the optimization. Therefore, DHLH performs best with a medium θ value. α is the parameter to balance the symmetric KL loss and the quantization loss. From Figure 8 (b) we can see that, the importance of pairwise supervision relationship is significantly reduced with a large α, leading to decreased performance. γ and λ control the sensitivity of the distance-to-probability function. As shown in Figures 8 (c) and (d), DHLH generally favors large γ and λ, because the magnitude of the exponential penalization is much larger in such cases, which helps to discriminate the fine-grained difference of the hard examples. On the other hand, too large γ or λ may lead to overfitting.

4 Conclusion

This paper introduces a novel deep hashing method, called dual hinge loss-based hashing (DHLH), for large-scale image retrieval. DHLH is an end-to-end deep hashing framework which well preserves the original data distribution in the Hamming space by the proposed dual hinge loss. The key idea underlying our method is that close similar/dissimilar image pairs, namely hard examples, conveys some critical information, e.g., their fine-grained difference, to be leveraged for more discriminative hashing. However, current hashing methods fail to effectively capture this information as their loss on easy examples (i.e., distant image pairs) often does not have finite upper bound, leading to the dominance of the easy examples in the optimization. The innovation our dual hinge loss contributes is its flexibility of effectively bounding penalization on the easy examples while at the same time having exponential penalization on the hard examples. This enables our method to learn not only the fine-grained difference of the hard examples but also the high-level proximity of the easy examples, resulting in very expressive hash codes. The effectiveness of our method is confirmed by its remarkable performance: (i) DHLH outperforms the best performer among eight state-of-the-art competing methods by a significant margin of 3.8%-8.7% in MAP, (ii) DHLH is able to use significantly shorter hash codes (i.e., 16 or 32 bits) to perform substantially better than, or comparably well to, the competing methods using 64-bit hash codes, and (iii) DHLH also achieves substantially better precision in the top-ranked returned results. We plan to extend the dual hinge loss idea to unsupervised deep hashing in our future work.

Acknowledgments

This work was supported by the National Natural Science Foundation of China project No. 61772057 and the support funding from State Key Lab. of Software Development Environment and Jiangxi and Qingdao Research Institute of Beihang University.

References

[1] Galen Andrew, Raman Arora, Jeff Bilmes, and Karen Livescu. 2013. Deep canonical correlation analysis. In *ICML*. 1247–1255.

[2] Xiao Bai, Cheng Yan, Haichuan Yang, Lu Bai, Jun Zhou, and Edwin Robert Hancock. 2018. Adaptive hash retrieval with kernel based similarity. *Pattern Recognition* 75 (2018), 136–148.

[3] Xiao Bai, Haichuan Yang, Jun Zhou, Peng Ren, and Jian Cheng. 2014. Data-dependent hashing based on p-stable distribution. *IEEE Transactions on Image Processing* 23, 12 (2014), 5033–5046.

[4] Yue Cao, Bin Liu, Mingsheng Long, Jianmin Wang, and MOE KLiss. 2018. Hash-GAN: Deep Learning to Hash with Pair Conditional Wasserstein GAN. In *CVPR*. 1287–1296.

[5] Yue Cao, Mingsheng Long, Bin Liu, Jianmin Wang, and MOE KLiss. 2018. Deep Cauchy Hashing for Hamming Space Retrieval. In *CVPR*. 1229–1237.

[6] Yue Cao, Mingsheng Long, Jianmin Wang, Qiang Yang, and Philip S. Yu. 2016. Deep Visual-Semantic Hashing for Cross-Modal Retrieval. In *SIGKDD*. 1445–1454.

[7] Zhangjie Cao, Mingsheng Long, Chao Huang, and Jianmin Wang. 2018. Transfer Adversarial Hashing for Hamming Space Retrieval. In *AAAI*. 6698–6075.

[8] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and S Yu Philip. 2017. HashNet: Deep Learning to Hash by Continuation.. In *ICCV*. 5609–5618.

[9] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 2009. NUS-WIDE: a real-world web image database from National University of Singapore. In *ICMR*. 48.

[10] Cheng Deng, Zhaojia Chen, Xianglong Liu, Xinbo Gao, and Dacheng Tao. 2018. Triplet-based deep hashing network for cross-modal retrieval. *IEEE Transactions on Image Processing* 27, 8 (2018), 3893–3903.

[11] Kun Ding, Bin Fan, Chunlei Huo, Shiming Xiang, and Chunhong Pan. 2017. Cross-Modal Hashing via Rank-Order Preserving. *IEEE Transactions on Multimedia* 19, 3 (2017), 571–585.

[12] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 12 (2013), 2916–2929.

[13] Jie Gui, Tongliang Liu, Zhenan Sun, Dacheng Tao, and Tieniu Tan. 2018. Fast supervised discrete hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 2 (2018), 490–496.

[14] Yanbin Hao, Tingting Mu, Richang Hong, Meng Wang, Ning An, and John Y Goulermas. 2017. Stochastic multiview hashing for large-scale near-duplicate video retrieval. *IEEE Transactions on Multimedia* 19, 1 (2017), 1–14.

[15] Kun He, Fatih Cakir, Sarah Adel Bargal, and Stan Sclaroff. 2018. Hashing as tie-aware learning to rank. In *CVPR*. 4023–4032.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

[17] Qing-Yuan Jiang and Wu-Jun Li. 2016. Deep cross-modal hashing. In *CVPR*. 3232–3240.

[18] Levent Karacan, Aykut Erdem, and Erkut Erdem. 2015. Image matting with KL-divergence based sparse sampling. In *ICCV*. 424–432.

[19] Alex Krizhevsky and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images. In *Techreport*.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In *NIPS*. 1097–1105.

[21] Brian Kulis and Trevor Darrell. 2009. Learning to hash with binary reconstructive embeddings. In *NIPS*. 1042–1050.

[22] Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. 2015. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*. 3270–3278.

[23] Kai Li, Guo Jun Qi, Jun Ye, and Kien A. Hua. 2017. Linear Subspace Ranking Hashing for Cross-Modal Retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 9 (2017), 1825–1838.

[24] Wu Jun Li, Sheng Wang, and Wang Cheng Kang. 2016. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*. 1711–1717.

[25] Jie Lin, Zechao Li, and Jinhui Tang. 2017. Discriminative Deep Hashing for Scalable Face Image Retrieval. In *IJCAI*. 2266–2272.

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*. 740–755.

[27] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. 2015. Deep hashing for compact binary codes learning. In *CVPR*. 2475–2483.

[28] Li Liu, Fumin Shen, Yuming Shen, Xianglong Liu, and Ling Shao. 2017. Deep sketch hashing: Fast free-hand sketch-based image retrieval. In *CVPR*. 2862–2871.

[29] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *CVPR*. 2074–2081.

[30] Xianglong Liu, Cheng Deng, Bo Lang, Dacheng Tao, and Xuelong Li. 2015. Query-adaptive reciprocal hash tables for nearest neighbor search. *IEEE Transactions on Image Processing* 25, 2 (2015), 907–919.

[31] Xianglong Liu, Junfeng He, and Shih-Fu Chang. 2017. Hash bit selection for nearest neighbor search. *IEEE Transactions on Image Processing* 26, 11 (2017), 5367–5380.

[32] Xianglong Liu, Lei Huang, Cheng Deng, Bo Lang, and Dacheng Tao. 2016. Query-adaptive hash code ranking for large-scale multi-view visual search. *IEEE Transactions on Image Processing* 25, 10 (2016), 4514–4524.

[33] Xianglong Liu, Lei Huang, Cheng Deng, Jiwen Lu, and Bo Lang. 2015. Multi-view complementary hash tables for nearest neighbor search. In *CVPR*. 1107–1115.

[34] Xianglong Liu, Zhujin Li, Cheng Deng, and Dacheng Tao. 2017. Distributed adaptive binary quantization for fast nearest neighbor search. *IEEE Transactions on Image Processing* 26, 11 (2017), 5324–5336.

[35] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised discrete hashing. In *CVPR*. 37–45.

[36] Fumin Shen, Yan Xu, Li Liu, Yang Yang, Zi Huang, and Heng Tao Shen. 2018. Unsupervised Deep Hashing with Similarity-Adaptive and Discrete Optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, 12 (2018), 3034–3044.

[37] Fumin Shen, Yang Yang, Li Liu, Wei Liu, Dacheng Tao, and Heng Tao Shen. 2017. Asymmetric binary coding for image search. *IEEE Transactions on Multimedia* 19, 9 (2017), 2022–2032.

[38] Antonio Torralba, Rob Fergus, and Yair Weiss. 2008. Small codes and large image databases for recognition. In *CVPR*. 1–8.

[39] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2012. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34, 12 (2012), 2393–2406.

[40] Miao Xie, Jiankun Hu, Song Guo, and Albert Y Zomaya. 2017. Distributed Segment-Based Anomaly Detection With Kullback-Leibler Divergence in Wireless Sensor Networks. *IEEE Transactions on Information Forensics and Security* 12, 1 (2017), 101–110.

[41] X. Xu, F. Shen, Y. Yang, H. T. Shen, and X. Li. 2017. Learning Discriminative Binary Codes for Large-scale Cross-modal Retrieval. *IEEE Transactions on Image Processing* 26, 5 (2017), 2494–2507.

[42] Haichuan Yang, Xiao Bai, Jun Zhou, Peng Ren, Zhihong Zhang, and Jian Cheng. 2014. Adaptive object retrieval with kernel reconstructive hashing. In *CVPR*. 1947–1954.

[43] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. 2016. Deep Hashing Network for efficient similarity retrieval. In *AAAI*. 2415–2421.

[44] Bohan Zhuang, Guosheng Lin, Chunhua Shen, and Ian Reid. 2016. Fast training of triplet-based deep binary embedding networks. In *CVPR*. 5955–5964.