Robust deep supervised hashing for image retrieval

Mo, Zhaoguo, Zhu, Yuesheng, Zhan, Jiawei

Zhaoguo Mo, Yuesheng Zhu, Jiawei Zhan, "Robust deep supervised hashing for image retrieval," Proc. SPIE 11519, Twelfth International Conference on Digital Image Processing (ICDIP 2020), 115191T (12 June 2020); doi: 10.1117/12.2573083

Event: Twelfth International Conference on Digital Image Processing, 2020, Osaka, Japan

Robust Deep Supervised Hashing for Image Retrieval

Zhaoguo Mo, Yuesheng Zhu, Jiawei Zhan
Communication and Information Security Laboratory, Shenzhen Graduate School, Peking University, Shenzhen 518055, People's Republic of China

ABSTRACT

Hashing is an important technique branch of image retrieval due to its satisfactory retrieval performance with high retrieval speed and low storage cost. Deep supervised hashing methods, which take advantage of the convolutional neural network and the supervised information, have shown better performance than other kinds of hashing methods. However, previous deep hashing methods do not consider the noisy data, which generally exist in large-scale labeled datasets and mislead the learning algorithm. In this paper, we propose a novel robust deep supervised hashing (RDSH) method, in which a robust pairwise loss and a quantitation loss are used to supervise the learning procedure. The quantitation loss guides the CNN to output binary codes. The robust pairwise loss for similarity-preserving learning is designed based on generalizations of the exponential and logarithmic functions. By adjusting its parameters, the robust pairwise loss can exhibit special properties, including tail-heaviness and boundedness, which results in the robustness of the learning procedure to noisy training data. To verify the robustness of the RDSH, we conduct experiments on CIFAR-10 with different noisy levels, in which RDSH shows better robustness than other deep supervised hashing methods. Experiments on standard CIFAR-10 and NUS-WIDE datasets show that RDSH outperforms other baselines.

Keywords: Hashing, deep learning, image retrieval

1. INTRODUCTION

Owing to the rapid growth of image data, efficient image retrieval in large-scale image databases becomes more and more important and urgent. Image retrieval is essentially a Nearest Neighbor (NN)[1] search problem. However, exact retrieval requires too much computing and storage resources to be practical. Approximate Nearest Neighbor (ANN) search, which tries to balance costs and retrieval accuracy, is a more practical choice. Hashing is an effective way to achieve ANN search. Hashing aims to encode high-dimensional data into low-dimensional binary codes using hash functions, which preserve the similarity in the original space. Retrieval with hashing code significantly reduces the computing cost and storage cost. Therefore, hashing has received increasing attention in recent years.

Locality sensitive hashing (LSH)[2] is the earliest hashing algorithm. LSH and its variants[3,4] generate hash codes using well-designed random projection functions, which project adjacent data into similar binary codes with high probability. LSH is a representative method of data-independent methods, whose hash functions are independent of the data. Data-independent methods usually need long hash codes to achieve the desired performance, leading to a degradation of the retrieval efficiency. Data-dependent methods that learn hash functions according to the data have shown better performance than data-independent methods.

Data-dependent methods can be divided into unsupervised hashing methods and supervised hashing methods. Unsupervised hashing methods learn hash functions utilizing unlabeled data. Representative unsupervised hashing methods include spectral hashing (SH)[5], iterative quantization (ITQ)[6], double-bit quantization (DBQ)[7] and anchor graph hashing (AGH)[8]. Supervised hashing methods learn hash functions utilizing feature information as well as label information of the data points, which outperform unsupervised hashing methods. Representative supervised hashing methods include binary reconstructive embedding (BRE)[9], supervised hashing with kernels (KSH)[10], similarity preserving algorithm for entropy-based coding (SpecH)[11], fast supervised hashing (FastH)[12], supervised discrete hashing (SDH)[13] and column sampling based discrete supervised hashing (COSDISH)[14].

Convolutional neural network (CNN) has been proven to be powerful in feature extraction for image data. Therefore, some deep supervised hashing methods, which use CNN for both image feature extraction and hash function learning, have been proposed and outperform traditional methods that use hand-crafted features for hash coding. Representative deep supervised hashing methods include convolutional neural network hashing (CNNH)[15], network in network hashing (NINH)[16], deep pairwise-supervised hashing (DPSH)[17], deep hashing network (DNH)[18], deep supervised discrete

Twelfth International Conference on Digital Image Processing (ICDIP 2020), edited by Xudong Jiang, Hiroshi Fujita
Proc. of SPIE Vol. 11519, 115191T · © 2020 SPIE · CCC code: 0277-786X/20/$21 · doi: 10.1117/12.2573083

hashing (DSDH)[19], hashing with mutual information (MIH)[20] and hashing as tie-aware learning to rank (TALR)[21]. For supervised hashing methods, supervised information (label) guides the learning procedure. However, the quality of the supervised information may decrease due to many factors, including low resolution, the small size of the object, the cover over the object, subjective annotation, mislabeling, and so on. These reasons broaden the gap between the semantic labels and the information actually expressed by the images. These noisy data will mislead the model during the learning procedure. Existing hashing methods do not consider the problem of noisy training data and perform greedily learning from the outliers, which result in sub-optimal hash functions.

In this paper, we propose a robust deep supervised hashing method (RDSH) to solve the problem of noisy training data. In RDSH, a CNN is used for simultaneously feature learning and hash function learning. The learning procedure is supervised by a robust pairwise loss and a quantitation loss. The quantitation loss guides the CNN to output binary codes. The robust pairwise loss for similarity-preserving learning is designed based on generalizations of the exponential and logarithmic functions. By adjusting its parameters, the robust pairwise loss can exhibit special properties, including tail-heaviness and boundedness, which enables it to control the impact of noisy data and normal data. By reducing the negative impact of noisy data and increasing the impact of normal data, the robust pairwise loss shows robustness to the noisy data. We conduct extensive experiments on CIFAR-10 and NUS-WIDE datasets to verify the robustness and effectiveness of RDSH. The experimental results demonstrate that RDSH shows better robustness than other baselines at different noisy levels. We further study how the parameters of the robust pairwise loss affect the robustness. Experiments on standard CIFAR-10 and NUS-WIDE datasets show that RDSH outperforms other baselines.

2. ROBUST DEEP SUPERVISED HASHING

2.1 Problem definition

While solving the problem of similar images retrieval using a supervised hashing method, we are given a training set of n images $\{\boldsymbol{x}_i\}_{i=1}^n$ along with a set of pairwise labels $S = \{s_{ij}\}$. $\boldsymbol{x}_i$ is the feature vector (hand-crafted feature or raw pixels) of image i and $s_{ij} \in \{-1,1\}$ is the similarity of the image pair (i,j), where $s_{ij} = 1$ means that image i and j are similar and $s_{ij} = -1$ means that image i and j are dissimilar. s_{ij} is usually generated according to the semantic labels of image pair (i,j). Our goal is to learn a hash function f that can map the feature $\boldsymbol{x}$ of an image into a binary code $\boldsymbol{b} = f(\boldsymbol{x}) \in \{-1,1\}^c$, where c is the length of the code. The codes $\boldsymbol{B} = \{\boldsymbol{b}_i\}_{i=1}^n$ generated by f are required to persist the semantic similarity of the image pairs. That is to say, the binary codes $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ are expected to take a small Hamming distance if the image pair (i,j) is similar and take a large Hamming distance if the image pair (i,j) is dissimilar.

2.2 Proposed method

In this paper, we propose a robust deep supervised hashing method (RDSH). RDSH provides two parameters to control the sensitivity of the learning procedure to noisy training data, which improves the robustness of RDSH to noisy training data. As shown in Figure 1, RDSH is divided into three parts, including an input module, a convolutional neural network (CNN) module, and a loss function module.

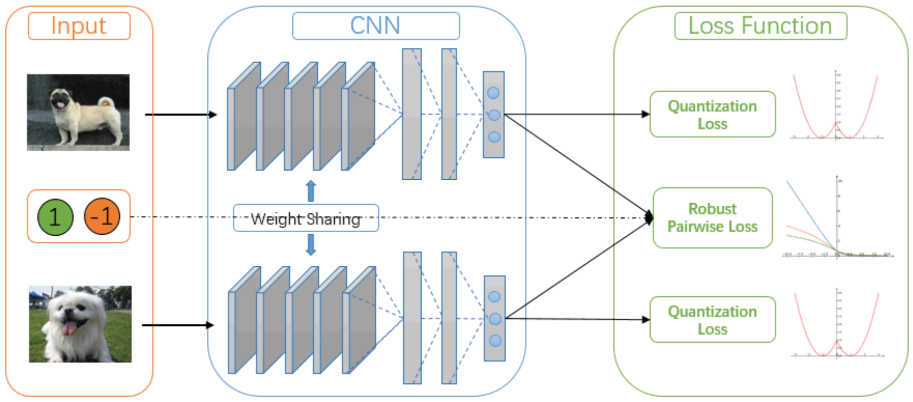


Figure 1. The architecture of the robust deep supervised hashing (RDSH).

The input module accepts the feature vectors of an image pair and their similarity label $(\boldsymbol{x}_i, \boldsymbol{x}_j, s_{ij})$ as the input of RDSH. The feature vectors of the image pair are their raw pixels, which will be resized to the input size of the CNN and then fed into two weight sharing CNNs respectively. The similarity label will be used for calculating the loss and updating the CNN parameters.

The CNN module, comprised of convolutional layers and fully-connected layers, plays a role in both feature extraction and hash function learning. Common CNNs, such as CNN-F[22], AlexNet[23], and ResNet[24], can be used in the CNN module. Without loss of generality, we use the CNN-F, which is the most popular one in the hashing community[17,19,20,21]. To generate hash codes with c bits, the last fully-connected layer of CNN-F is replaced by a new fully-connected layer with c neurons. We adopt identity function as the activation function for the new fully-connected layer. The CNN's output of image i is denoted as $\boldsymbol{r}_i = \phi(\boldsymbol{x}_i; \boldsymbol{\theta}) \in \mathbb{R}^c$, where $\boldsymbol{\theta}$ represents the parameters of the CNN. While training the CNN, we directly use the output of the CNN to calculate the losses and their derivatives. After the training, we apply an element-wise sign function $\mathrm{sign}(\cdot)$ on the output of the CNN to generate binary code, where $\mathrm{sign}(r)$ returns 1 if $r > 0$ and returns -1 if $r \leq 0$. More concretely, the hash code of image i is generated by calculating $\boldsymbol{b}_i = \mathrm{sign}(\phi(\boldsymbol{x}_i; \boldsymbol{\theta})) \in \{-1,1\}^c$.

The loss function module includes a robust pairwise loss and a quantization loss. The robust pairwise loss is applied on the CNN's output of an images pair, which can preserve the similarity of the image pair even though the training set contains noise. The robust pairwise loss will be derived later. Our goal is to generate binary codes for images when the output of the CNN is a real vector. To bridge the gap, an intuitive way is to apply an element-wise sign function on the output of the CNN. But this operation will lose a lot of information in the real vector. So we apply a quantization loss on the CNN's output of every single image to guide the CNN to generate output close to -1 or 1 spontaneously, which narrows the gap and reduces the loss of information. The quantization loss used in this paper is defined as

$$\mathcal{L}_q = \sum_{i=1}^{n} \|\boldsymbol{r}_i - \mathrm{sign}(\boldsymbol{r}_i)\|_2^2 = \sum_{i=1}^{n} \sum_{k=1}^{c} \left(\boldsymbol{r}_i^k - \mathrm{sign}(\boldsymbol{r}_i^k)\right)^2, \tag{1}$$

where $\|\cdot\|_2$ is L2 norm of a vector, $\boldsymbol{r}_i^k$ is the kth dimension data of $\boldsymbol{r}_i$. As shown in Figure 2, the quantization loss of an element of the CNN's output increases rapidly as it moves away from -1 or 1 and gradually returns to zero otherwise.

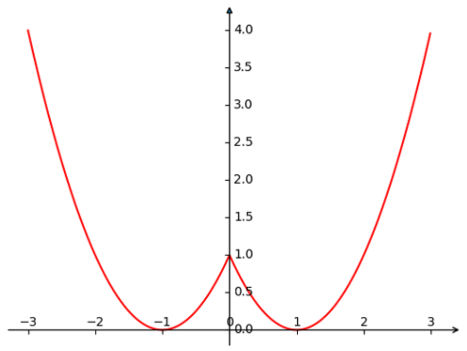


Figure 2. The quantization loss of an element of the CNN's output.

- **Logistic Regression Pairwise Loss**

Given hash codes $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$, there is a linear relationship between their Hamming distance $d_{ham}(\boldsymbol{b}_i, \boldsymbol{b}_j)$ and inner product ${\boldsymbol{b}_i}^T\boldsymbol{b}_j$: $d_{ham}(\boldsymbol{b}_i, \boldsymbol{b}_j) = \frac{1}{2}(c - {\boldsymbol{b}_i}^T\boldsymbol{b}_j)$. So while training, the inner product of $\boldsymbol{r}_i$ and $\boldsymbol{r}_j$ is used to quantify the similarity of the image pair (i, j), denoted as $\alpha_{ij} = \beta {\boldsymbol{r}_i}^T \boldsymbol{r}_j$. The parameter $\beta > 0$ can adjust the range of α_{ij}, which is important to the robust pairwise loss and will be discussed further later. Given the CNN's output for all images in the training set $\boldsymbol{R} = \{\boldsymbol{r}_i\}_{i=1}^n$, the likelihood of the similarity labels $s_{ij} \in S$ is defined as

$$p\left(s_{ij} \middle| \boldsymbol{r}_i, \boldsymbol{r}_j\right) = \exp\left(s_{ij}\frac{\alpha_{ij}}{2} - G\left(\frac{\alpha_{ij}}{2}\right)\right), \tag{2}$$

where $G(\alpha_{ij}/2)$ guarantees $p(1|\boldsymbol{r}_i, \boldsymbol{r}_j) + p(-1|\boldsymbol{r}_i, \boldsymbol{r}_j) = 1$, i.e.

$$\exp\left(\frac{\alpha_{ij}}{2} - G\left(\frac{\alpha_{ij}}{2}\right)\right) + \exp\left(-\frac{\alpha_{ij}}{2} - G\left(\frac{\alpha_{ij}}{2}\right)\right) = 1.$$

By multiplying both sides of this equation by $\exp\left(G\left(\frac{\alpha_{ij}}{2}\right)\right)$ and taking their logarithm, we get

$$G\left(\frac{\alpha_{ij}}{2}\right) = \log\left(\exp\left(\frac{\alpha_{ij}}{2}\right) + \exp\left(-\frac{\alpha_{ij}}{2}\right)\right). \tag{3}$$

While $s_{ij} = 1$, $p\left(s_{ij}|\boldsymbol{r}_i, \boldsymbol{r}_j\right) = 1/\left(1 + \exp(-\alpha_{ij})\right)$ is a sigmoid function with respect to the pairwise similarity α_{ij}. The larger the inner product $\boldsymbol{r}_i{}^T\boldsymbol{r}_j$ is, the larger the conditional probability $p\left(s_{ij}|\boldsymbol{r}_i, \boldsymbol{r}_j\right)$ will be, meaning that the image pair (i, j) should be similar with greater confidence. While $s_{ij} = -1$, we get $p\left(s_{ij}|\boldsymbol{r}_i, \boldsymbol{r}_j\right) = 1/\left(1 + \exp(\alpha_{ij})\right)$. The smaller the inner product $\boldsymbol{r}_i{}^T\boldsymbol{r}_j$ is, the larger the conditional probability $p\left(s_{ij}|\boldsymbol{r}_i, \boldsymbol{r}_j\right)$ will be, meaning that the image pair (i, j) should be dissimilar with greater confidence. Hence, $p\left(s_{ij}|\boldsymbol{r}_i, \boldsymbol{r}_j\right)$ is a reasonable likelihood to bridge the CNN's output of the image pair $(\boldsymbol{r}_i, \boldsymbol{r}_j)$ and their similarity label s_{ij}.

The likelihood of all the similarity labels S is defined as

$$p(S|\boldsymbol{R}) = \prod_{s_{ij} \in S} p\left(s_{ij}|\boldsymbol{r}_i, \boldsymbol{r}_j\right). \tag{4}$$

By taking the negative logarithm of $p(S|\boldsymbol{R})$, we get the following pairwise loss

$$\begin{aligned}\mathcal{L}_p = -\log p(S|\boldsymbol{R}) &= \sum_{s_{ij} \in S} -\log p\left(s_{ij}|\boldsymbol{r}_i, \boldsymbol{r}_j\right) \\ &= \sum_{s_{ij} \in S} -\log \exp\left(s_{ij}\frac{\alpha_{ij}}{2} - G\left(\frac{\alpha_{ij}}{2}\right)\right).\end{aligned} \tag{5}$$

Equation (5) is essentially a logistic regression loss based on the pairwise similarity α_{ij}.

- **Robust Pairwise Loss**

Logistic regression pairwise loss learns from outliers greedy, which makes it sensitive to noisy samples. That means the noisy samples in the training set will mislead the optimization direction for CNN. So the CNN trained with noisy data using a logistic regression pairwise loss is biased, resulting in the feature and hash codes generated by the CNN is suboptimal. To increase the robustness of pairwise loss, we introduce a generalized exponential function $\exp_t$ and a logarithmic function $\log_t$[25].

For $t > 0$, $\exp_t$ is defined as

$$\exp_t(x) = \begin{cases} \exp(x) & t = 1 \\ [1 + (1-t)x]_+^{\frac{1}{(1-t)}} & t \neq 1 \end{cases}, \tag{6}$$

where $[\cdot]_+ = \max(\cdot, 0)$. The graphs of generalized exponential function with different t are shown in Figure 3(a). $\exp_t$ has a heavy tail for $t > 1$ and recovers to the standard exponential function in the limit $t \to 1$. The derivative of $\exp_t$ is

$$\frac{\partial \exp_t(x)}{\partial \mathrm{x}} = \exp_t(x)^t. \tag{7}$$

For $t > 0$, $\log_t$ is defined as

$$\log_t(x) = \begin{cases} \log(x) & t = 1 \\ \dfrac{1}{(1-t)}(x^{1-t} - 1) & t \neq 1. \end{cases} \tag{8}$$

The graphs of generalized logarithmic function with different t are shown in Figure 3(b). $\log_t$ recovers to the standard logarithmic function in the limit $t \to 1$. $\log_t$ is lower bonded by $-1/(1-t)$ for $0 < t < 1$. The derivative of $\log_t$ is

$$\frac{\partial \log_t(x)}{\partial \mathrm{x}} = \frac{1}{x^t}. \tag{9}$$

We replace the standard exponential function exp of the likelihood function in Equation (2) with $\exp_{t_2}$. Correspondingly, we take the negative logarithm of $p(S|\boldsymbol{R})$ using the generalized logarithmic function $\log_{t_1}$. Then we get the robust pairwise loss:

$$\begin{aligned} \mathcal{L}_{rp} = -\log_{t_1} p_{t_2}(S|\boldsymbol{R}) &= \sum_{s_{ij} \in S} -\log_{t_1} p_{t_2}(s_{ij}|\boldsymbol{r}_i, \boldsymbol{r}_j) \\ &= \sum_{s_{ij} \in S} -\log_{t_1} \exp_{t_2}\left(s_{ij}\frac{\alpha_{ij}}{2} - G_{t2}\left(\frac{\alpha_{ij}}{2}\right)\right). \end{aligned} \tag{10}$$

Similar to Equation (2), $G_{t2}\left(\frac{\alpha_{ij}}{2}\right)$ is used to guarantee

$$p_{t_2}(1|\boldsymbol{r}_i, \boldsymbol{r}_j) + p_{t_2}(-1|\boldsymbol{r}_i, \boldsymbol{r}_j) = 1. \tag{11}$$

But for $t_2 \neq 1$, $G_{t2}\left(\frac{\alpha_{ij}}{2}\right)$ does not have a closed-form solution like Equation (3). To calculate $G_{t2}\left(\frac{\alpha_{ij}}{2}\right)$ numerically, Zhang et al.[25] provides an iterative method suitable for $\alpha_{ij} \geq 0$. From Equation (11), we can easily get that $G_{t_2}(-\alpha/2) = G_{t_2}(\alpha/2)$. So we provide the Algorithm 1 for calculating $G_{t_2}(\alpha/2)$ for $\alpha_{ij} \in$ R.

Algorithm 1 Iterative algorithm for calculating $G_{t_2}(\alpha/2)$ in $\mathcal{L}_{rp}$

```
**Input:** $\alpha \in$ R.
**Output:** $G_{t_2}(\alpha/2)$.
$\tilde{\alpha} \leftarrow |\alpha|$.
**while** $\tilde{\alpha}$ *is not converged* **do**
    $Z(\tilde{\alpha}) \leftarrow 1 + \exp_{t2}(-\tilde{\alpha})$.
    $\tilde{\alpha} \leftarrow Z(\tilde{\alpha})^{1-t}|\alpha|$.
**end**
$G_{t_2}(\alpha/2) \leftarrow -\log_{t2}\left(\frac{1}{Z(\tilde{\alpha})}\right) + \frac{|\alpha|}{2}$.
```

$\mathcal{L}_{rp}$ provides two parameters t_1 and t_2 to control the sensitivity of the hashing learning algorithm to outliers. More specifically, while $s_{ij} = 1$, the graphs of the robust pairwise loss with different t_1 and t_2 are shown in Figure 4. In Figure 4(a), we fix $t_2 = 1.0$ and take different values for $t_1 \leq t_2$. As t_1 decreases, the loss of a similar image pair (i, j) with small inner product decreases. In fact, when $0 < t_1 < 1$, $\mathcal{L}_{rp}$ is lower bonded by $-1/(1-t_1)$. These properties can be used to limit the impact of noise samples that are similar but take dissimilar labels. In Figure 4(b), we fix $t_1 = 1.0$ and take different values for $t_2 \geq 0$. As t_2 increases, the loss of a similar image pair (i, j) with small inner product decreases. At the same time, the loss of a similar image pair (i, j) with large inner product increases. The former property can limit the impact of noise samples while the latter one can enhance the normal samples. We can get similar properties of $\mathcal{L}_{rp}$ for $s_{ij} = -1$. In the experiment, we will try more combination of (t_1, t_2) to observe how they affect the result.

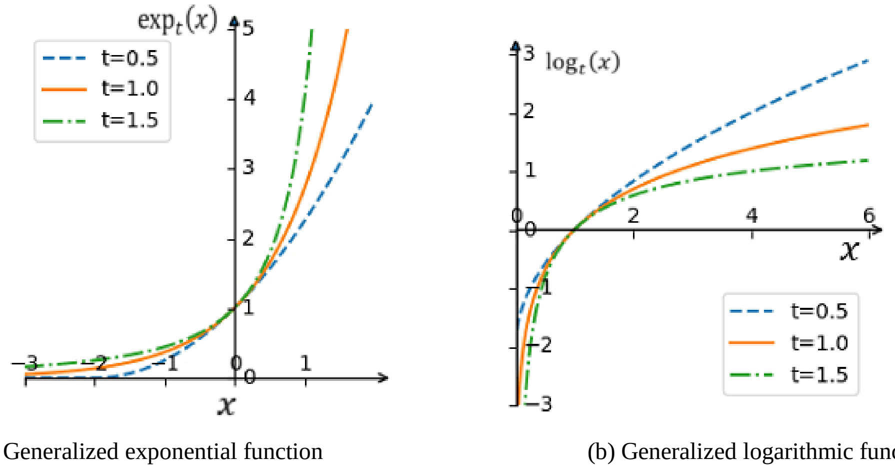


Figure 3. Generalized exponential function and generalized logarithmic function with different t.

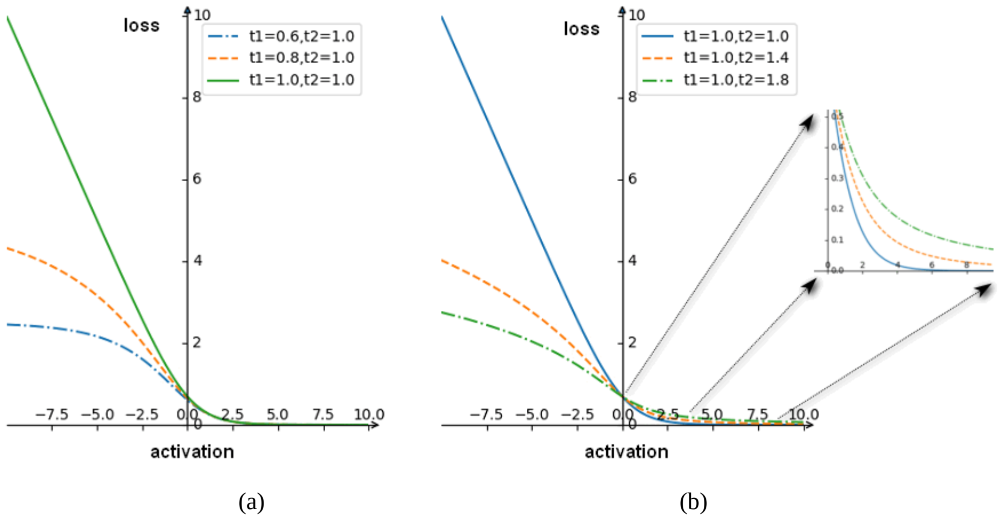


Figure 4. Robust pairwise loss with different t_1 and t_2.

Different settings of the hash code's length c will change the range of α_{ij}. Under the limitation of the quantization loss, the output of the CNN $\boldsymbol{r}$ is close to a binary vector belongs to $\{-1,1\}^c$. So for the given c and β, the range of α_{ij} is approximate to $[-c\beta, c\beta]$. When β is fixed, the range of α_{ij} is proportional to c. Therefore, a small value of c results in a small range of α_{ij}, which makes it more difficult to adjust the robust pairwise loss. To keep a stable range of α_{ij}, we take different values for β under different settings of c.

By taking Equation (1) and (10) together, we achieve the following optimization problem:

$$\min_{\boldsymbol{\theta}} \mathcal{L}_a = \mathcal{L}_{rp} + \eta \mathcal{L}_q$$
$$= \sum_{s_{ij} \in S} -\log_{t_1} \exp_{t_2}\left(s_{ij}\frac{\alpha_{ij}}{2} - G_{t2}\left(\frac{\alpha_{ij}}{2}\right)\right) + \eta \sum_{i=1}^{n} \|\boldsymbol{r}_i - \mathrm{sign}(\boldsymbol{r}_i)\|_2^2, \tag{12}$$

where η is the weight parameter of the quantization loss.

2.3 Learning

The proposed method uses CNN for feature extraction and hash coding in an end-to-end fashion. Therefore, the only parameter to be optimized is the parameters $\boldsymbol{\theta}$ of the CNN. We will derive the derivative of $\mathcal{L}_a$ with respect to $\boldsymbol{r}_i$. Then $\boldsymbol{\theta}$ can be updated utilizing backpropagation.

We firstly focus on the derivative of $\mathcal{L}_{rp}$. Since $G_{t_2}(\alpha/2)$ does not have a closed-form solution, we cannot solve its derivative directly. As shown in Theorem 1, we derive it in another way.

Theorem 1 For probability distributions

$$p_t(s|\alpha) = \exp_t\left(s\frac{\alpha}{2} - G\left(\frac{\alpha}{2}\right)\right),$$

Where $s \in \{-1,1\}$ and $\sum_{\varepsilon\in\{-1,1\}} p_t(\varepsilon|\alpha) = 1$. If

$$\frac{\partial}{\partial\alpha}\sum_{\varepsilon\in\{-1,1\}} p_t(\varepsilon|\alpha) = \sum_{\varepsilon\in\{-1,1\}} \frac{\partial}{\partial\alpha} p_t(\varepsilon|\alpha), \tag{13}$$

then

$$\frac{\partial G\left(\frac{\alpha}{2}\right)}{\partial\alpha} = \frac{1}{2}\frac{\sum_{\varepsilon\in\{-1,1\}} \varepsilon p_t(\varepsilon|\alpha)^t}{\sum_{\varepsilon\in\{-1,1\}} p_t(\varepsilon|\alpha)^t}. \tag{14}$$

Proof Since $\sum_{\varepsilon\in\{-1,1\}} p_t(\varepsilon|\alpha) = 1$, we can get from Equation (13) that

$$0 = \sum_{\varepsilon\in\{-1,1\}} \frac{\partial}{\partial\alpha} p_t(\varepsilon|\alpha) = \sum_{\varepsilon\in\{-1,1\}} \frac{\partial}{\partial\alpha}\exp_t\left(\varepsilon\frac{\alpha}{2} - G\left(\frac{\alpha}{2}\right)\right)$$
$$0 = \sum_{\varepsilon\in\{-1,1\}} \exp_t\left(\varepsilon\frac{\alpha}{2} - G\left(\frac{\alpha}{2}\right)\right)^t \left(\frac{\varepsilon}{2} - \frac{\partial G\left(\frac{\alpha}{2}\right)}{\partial\alpha}\right)$$
$$0 = \sum_{\varepsilon\in\{-1,1\}} \frac{\varepsilon}{2}\exp_t\left(\varepsilon\frac{\alpha}{2} - G\left(\frac{\alpha}{2}\right)\right)^t - \frac{\partial G\left(\frac{\alpha}{2}\right)}{\partial\alpha}\sum_{\varepsilon\in\{-1,1\}} \exp_t\left(\varepsilon\frac{\alpha}{2} - G\left(\frac{\alpha}{2}\right)\right)^t$$

then

$$\frac{\partial G\left(\frac{\alpha}{2}\right)}{\partial\alpha} = \frac{\sum_{\varepsilon\in\{-1,1\}}\frac{\varepsilon}{2}\exp_t\left(\varepsilon\frac{\alpha}{2} - G\left(\frac{\alpha}{2}\right)\right)^t}{\sum_{\varepsilon\in\{-1,1\}}\exp_t\left(\varepsilon\frac{\alpha}{2} - G\left(\frac{\alpha}{2}\right)\right)^t} = \frac{1}{2}\frac{\sum_{\varepsilon\in\{-1,1\}} \varepsilon p_t(\varepsilon|\alpha)^t}{\sum_{\varepsilon\in\{-1,1\}} p_t(\varepsilon|\alpha)^t}.$$

For the sake of simplicity, we define the robust pairwise loss with a single image pair as

$$\mathcal{L}_{ij} = -\log_{t_1} \exp_{t_2}\left(s_{ij}\frac{\alpha_{ij}}{2} - G_{t2}\left(\frac{\alpha_{ij}}{2}\right)\right). \tag{15}$$

Using Equation (7), (9) and (14), we can get the derivative of $\mathcal{L}_{ij}$ with respect to $\boldsymbol{r}_i$

$$\begin{aligned}\frac{\partial \mathcal{L}_{ij}}{\partial \boldsymbol{r}_i} &= \frac{\partial}{\partial \alpha_{ij}}\left(-\log_{t_1} \exp_{t_2}\left(s_{ij}\frac{\alpha_{ij}}{2} - G_{t2}\left(\frac{\alpha_{ij}}{2}\right)\right)\right)\frac{\partial \alpha_{ij}}{\partial \boldsymbol{r}_i} \\ &= -\beta \exp_{t_2}\left(s_{ij}\frac{\alpha_{ij}}{2} - G_{t2}\left(\frac{\alpha_{ij}}{2}\right)\right)^{t_2-t_1}\left(\frac{s_{ij}}{2} - \frac{\partial}{\partial \alpha_{ij}}G_{t2}\left(\frac{\alpha_{ij}}{2}\right)\right)\boldsymbol{r}_j \\ &= -\frac{\beta}{2}p_{t_2}\left(s_{ij}\middle|\boldsymbol{r}_i, \boldsymbol{r}_j\right)^{t_2-t_1}\left(s_{ij} - \frac{\sum_{\varepsilon\in\{-1,1\}}\varepsilon p_{t_2}\left(\varepsilon\middle|\boldsymbol{r}_i, \boldsymbol{r}_j\right)^{t_2}}{\sum_{\varepsilon\in\{-1,1\}}p_{t_2}\left(\varepsilon\middle|\boldsymbol{r}_i, \boldsymbol{r}_j\right)^{t_2}}\right)\boldsymbol{r}_j.\end{aligned} \tag{16}$$

Similarly,

$$\frac{\partial \mathcal{L}_{ij}}{\partial \boldsymbol{r}_j} = -\frac{\beta}{2}p_{t_2}\left(s_{ij}\middle|\boldsymbol{r}_i, \boldsymbol{r}_j\right)^{t_2-t_1}\left(s_{ij} - \frac{\sum_{\varepsilon\in\{-1,1\}}\varepsilon p_{t_2}\left(\varepsilon\middle|\boldsymbol{r}_i, \boldsymbol{r}_j\right)^{t_2}}{\sum_{\varepsilon\in\{-1,1\}}p_{t_2}\left(\varepsilon\middle|\boldsymbol{r}_i, \boldsymbol{r}_j\right)^{t_2}}\right)\boldsymbol{r}_i. \tag{17}$$

We can get the derivative of $\mathcal{L}_{rp}$ with respect to $\boldsymbol{r}_i$ based on Equation (16) and (17)

$$\frac{\partial \mathcal{L}_{rp}}{\partial \boldsymbol{r}_i} = \sum_{j:s_{ij}\in S}\frac{\partial \mathcal{L}_{ij}}{\partial \boldsymbol{r}_i} + \sum_{j:s_{ji}\in S}\frac{\partial \mathcal{L}_{ji}}{\partial \boldsymbol{r}_i}. \tag{18}$$

The derivative of $\mathcal{L}_q$ with respect to $\boldsymbol{r}_i$ is

$$\begin{aligned}\frac{\partial \mathcal{L}_q}{\partial \boldsymbol{r}_i} &= \frac{\partial}{\partial \boldsymbol{r}_i}\sum_{j=1}^{n}\left\|\boldsymbol{r}_j - \operatorname{sign}(\boldsymbol{r}_j)\right\|_2^2 \\ &= 2\left(\boldsymbol{r}_j - \operatorname{sign}(\boldsymbol{r}_j)\right).\end{aligned} \tag{19}$$

Then we get the derivative of $\mathcal{L}_a$ with respect to $\boldsymbol{r}_i$

$$\frac{\partial \mathcal{L}_a}{\partial \boldsymbol{r}_i} = \frac{\partial \mathcal{L}_{rp}}{\partial \boldsymbol{r}_i} + \eta\frac{\partial \mathcal{L}_q}{\partial \boldsymbol{r}_i}. \tag{20}$$

Using Equation (20), we can update the parameters of the CNN $\boldsymbol{\theta}$ utilizing backpropagation. The concrete learning algorithm of the proposed method is shown in Algorithm 2.

```
Algorithm 2 The learning algorithm of the proposed method
    **Input:** a set of images for training$\{\boldsymbol{x}_i\}_{i=1}^{n}$, similarity labels of the image pairs in training set $S = \{s_{ij}\}$,
     parameter $\beta$ and $\eta$.
    **Output:** the parameters of the CNN $\boldsymbol{\theta}$.
    Initialize the first seven layers of the CNN using the parameters of the pre-training CNN-F model; Initialize
    the last layer with a Gaussian distribution $\mathrm{N}(0, 0.1)$.
    **For** epoch=1,2,…,T **do**
    For iteration=1,2,…,n/k do
        Randomly sample $k$ images $\{\boldsymbol{x}_i\}_{i=1}^{k} \subset \{\boldsymbol{x}_i\}_{i=1}^{n}$, and $S_{batch} \subset S$ denotes their similarity labels.
        Calculate $\boldsymbol{r}_i \leftarrow \phi(\boldsymbol{x}_i; \boldsymbol{\theta})$ for each image $\boldsymbol{x}_i \in \{\boldsymbol{x}_i\}_{i=1}^{k}$.
        Calculate $G_{t2}\left(\frac{\alpha_{ij}}{2}\right)$ for each image pair $(i,j) \in S_{batch}$ using Algorithm 1.
        Calculate the derivative of $\mathcal{L}_a$ for each image $\boldsymbol{x}_i \in \{\boldsymbol{x}_i\}_{i=1}^{k}$ according to Equation (20).
        Update $\boldsymbol{\theta}$ utilizing backpropagation.
    end for
end for
```

3. EXPERIMENT

To evaluate RDSH and other baselines, we conduct extensive experiments on two benchmark datasets for image retrieval: CIFAR-10[26] and NUS-WIDE[27]. The proposed method is implemented with MatConvNet[28].

3.1 Experimental setup

CIFAR-10 contains 60,000 color images with a resolution of 32x32. Each image is labeled with one of ten classes, and each class contains 6,000 images. NUS-WIDE contains 269,548 web images collected from the Flicker website. Each image is labeled with one or multiple classes of 81 classes. Similar to previous work[19,17], we use a subset containing 195,834 images that belong to the 21 most frequent classes., where each class contains at least 5,000 images.

For both CIFAR-10 and NUS-WIDE, 100 images of each class are selected randomly to constitute the query set, and the remaining images constitute the retrieval set. For unsupervised methods, the whole retrieval set is used for training. For supervised methods, 500 images of each class are selected randomly from the retrieval set to constitute the training set. The class labels are used to construct the similarity label. A pair of images are seen as similar if they share at least one common class label. Otherwise, they are seen as dissimilar.

We use the mean average precision (MAP) to measure the accuracy of the hashing method. Following previous work[19,17], the MAP is calculated according to the top 5,000 returned neighbors for NUS-WIDE. The maximum epoch of RDSH is set as 60. The learning rate is 0.01 and reduces to 0.001 after 40 epochs. η is set as 0.1.

3.2 Robustness to noise

To verify the robustness of RDSH to noisy data in training set, we add extra noise to the training set of CIFAR-10 and compare the performance of RDSH against other deep supervised hashing methods, including DPSH[17], DSDH[19], TALR[21] and MIH[20]. We add noise by replacing the labels with random labels. For example, if we want to add 10% noise to the CIFAR-10, we will randomly sample 10% images from each class in the training set of CIFAR-10 and replace the labels of the sampled images with random labels. We use the clean query set and the clean retrieval set for testing. For RDSH, we try different compositions of (t_1, t_2) for $0 < t_1 < t_2 < 2.4$ and select the optimal compositions under different noise levels.

Table 1 shows the MAP of different methods trained at different noise levels in CIFAR-10 while the code length is 48. RDSH outperforms other methods at all noise levels. As the noise level increase, the MAP of RDSH decreases much slower than other methods, which indicates that RDSH is more robust to noise. The optimal compositions of (t_1, t_2) at 0%-50% noise levels are (2.0,2.6), (2.0,2.6), (0.8,1.2), (0.8,1.2), (0.6,1.2) and (0.6,1.2) respectively. At lower noise levels such as 0% and 10%, a large t_2 to get notable tail-heaviness and a t_1 less than t_2 with an appropriate gap get the best performance, because the tail-heaviness makes the algorithm pays more attention to the normal samples and a t_1 less than t_2 makes the algorithm focuses less on the noisy samples. At noise levels higher than 10%, a t_1 less than 1 to get boundedness of the loss with a t_2 slightly greater than 1 gets the best performance. Because the boundedness of the loss provides a more powerful limitation on the impact of the noisy samples.

Taking the 30% noise level as an example, we take a more careful observation of the relationship between the selection of (t_1, t_2) and the retrieval performance of RDSH. We assign different values to t_1 around 1 and take t_2 that not less than t_1 and 0. Table 2 shows the MAP using different compositions of (t_1, t_2) in 30% contaminated CIFAR-10 while the code length is 48. Underline emphasizes the best result with the same t_1 and boldface highlights the best result of all (t_1, t_2) compositions. While t_1 is fixed, the MAP increases first and then decreases as t_2 increases. That is because if t_2 is approximately larger than t_1, the algorithm focuses more on the normal samples and less on the outliers, which reduces the negative effect of noisy samples. However, if the gap between t_2 and t_1 is too large, the slope of the loss will be too small, which is not conducive to the convergence of the algorithm. By comparing the peak result at different t_1, we can further find that $t_1 < 0$ performs better than $t_1 \geq 0$. That is because while the noise level is high, the boundedness of the loss at $t_1 < 0$ can limit the negative effect of noisy samples more effectively.

Table 1. The MAP of different methods trained at different noise levels in CIFAR-10 (48bits).

Method	CIFAR-10					
Noise level	0%	10%	20%	30%	40%	50%
RDSH	**0.854**	**0.841**	**0.798**	**0.752**	**0.687**	**0.576**
TALR	0.810	0.746	0.660	0.599	0.540	0.331
MIH	0.796	0.765	0.683	0.515	0.483	0.272
DSDH	0.819	0.754	0.621	0.497	0.367	0.269
DPSH	0.752	0.661	0.575	0.435	0.349	0.271

Table 2. The MAP of RDSH using different compositions of (t_1, t_2) in 30% contaminated CIFAR-10 (48bits).

t_2 / t_1	t_1 +0	t_1 +0.2	t_1 +0.4	t_1 +0.6	t_1 +0.8	t_1 +1.0	t_1 +1.2
0.6	-	-	0.721	<u>0.748</u>	0.686	0.680	0.313
0.8	-	0.719	**<u>0.752</u>**	0.751	0.751	0.715	0.695
1.0	0.647	0.690	0.724	<u>0.737</u>	0.735	0.734	0.725
1.2	0.646	0.681	0.707	0.724	0.733	<u>0.735</u>	0.728

3.3 Result on CIFAR-10 and NUS-WIDE

Table 3. The MAP of different methods on CIFAR-10 and NUS-WIDE. The MAP on NUS-WIDE dataset is calculated based on the top 5,000 returned neighbors.

Method	CIFAR-10				Method	NUS-WIDE			
	12bits	24bits	32bits	48bits		12bits	24bits	32bits	48bits
RDSH	**0.810**	**0.837**	**0.838**	**0.854**	**RDSH**	**0.801**	**0.832**	**0.842**	**0.851**
MIHash	0.725	0.768	0.784	0.796	MIHash	0.776	<u>0.826</u>	0.832	<u>0.843</u>
TALR	<u>0.759</u>	0.790	<u>0.808</u>	0.810	TALR	<u>0.780</u>	0.804	<u>0.834</u>	0.833
DSDH	0.724	<u>0.792</u>	0.799	<u>0.822</u>	DSDH	0.779	0.810	0.821	0.829
DPSH	0.704	0.723	0.747	0.761	DPSH	0.771	0.800	0.808	0.819
FastH	0.305	0.349	0.369	0.384	FastH	0.621	0.65	0.665	0.687
SDH	0.285	0.329	0.341	0.356	SDH	0.568	0.600	0.608	0.637
KSH	0.303	0.337	0.346	0.356	KSH	0.556	0.572	0.581	0.588
ITQ	0.162	0.169	0.172	0.175	ITQ	0.452	0.468	0.472	0.477
SH	0.127	0.128	0.126	0.129	SH	0.454	0.406	0.405	0.400

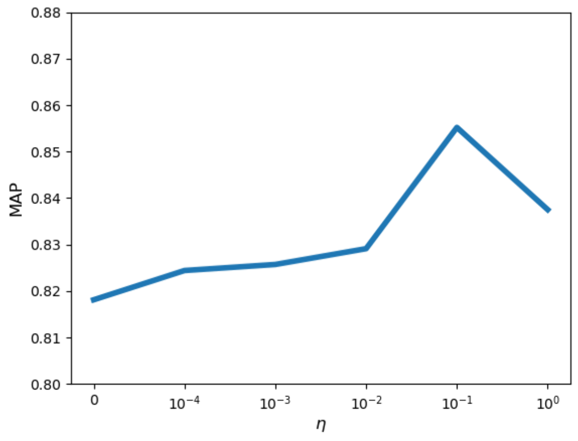
In this section, we compare RDSH with some baseline methods on two benchmark datasets: CIFAR-10 and NUS-WIDE. The compared methods include traditional unsupervised methods, traditional supervised methods, and deep hashing methods. Compared Traditional unsupervised methods include SH[5] and ITQ[6]. Compared Traditional supervised methods include KSH[10], SDH[13] and FastH[12]. For deep hashing methods, we use DPSH[17], DSDH[19], TALR[21] and MIH[20]. For traditional methods, we use the hand-crafted features, which include a 512-dimensional GIST descriptor to represent images of CIFAR-10 and an 1134-dimensional feature vector to represent images of NUS-WIDE. For deep hashing methods, we use the original pixels as input. For RDSH, (t_1, t_2) is set to $(2.0, 2.6)$ and $(0.8, 1.2)$ for CIFAR-10 and NUS-WIDE respectively.

In CIFAR-10, low resolution (32x32) of the images, the small size of the object, and the cover over the object may result in a gap between the semantic label and the actual semantics of an image. In NUS-WIDE, which is a large-scale multi-label dataset in real scenes, complicated background interference, subjective annotation, and mislabeling will also lead to some noisy data. These outliers will mislead methods that pay no attention to the robustness to noisy data. Table 3 shows the MAP of different methods on CIFAR-10 and NUS-WIDE. It is easy to find that our RDSH outperforms other methods, including traditional unsupervised methods, traditional supervised methods, and deep hashing methods.

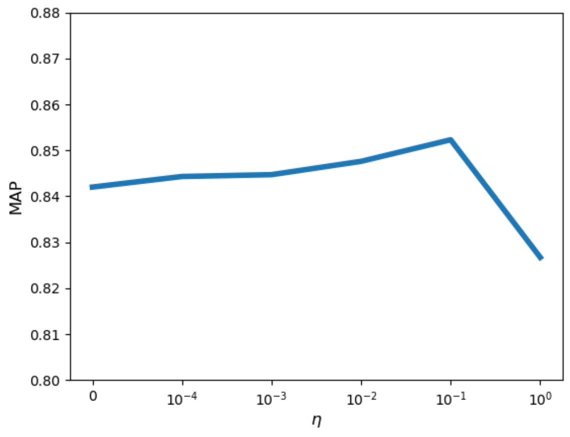
3.4 Discussion

The sensitivity of RDSH to η for CIFAR-10 and NUS-WIDE datasets is shown in Figure 5. We try different values for η between 0 and 1, and find that $\eta = 0.1$ achieved the best performance for both datasets. While not using the quantitation loss ($\eta = 0$), the MAP is less than that of $\eta = 0.1$.

In Table 4, we compare two different settings of β. RDSH (β=0.5) means we fix $\beta = 0.5$ when the length of the hash code is different. On the contrary, RDSH (adaptive) means we use $\beta = 0.5$ while $c = 48$, and use different β for other c to get a fixed range of α_{ij}. It is easy to find that RDSH (adaptive) outperforms RDSH (β=0.5) at small c.



(a) CIFAR-10



(b) NUS-WIDE

Figure 5. Sensitivity of RDSH to η for CIFAR-10 and NUS-WIDE (48bits).

Table 4. The MAP of different settings of β on CIFAR-10 and NUS-WIDE. The MAP on NUS-WIDE dataset is calculated based on the top 5,000 returned neighbors.

Method	CIFAR-10				Method	NUS-WIDE			
	12bits	24bits	32bits	48bits		12bits	24bits	32bits	48bits
RDSH (adaptive)	**0.810**	**0.837**	**0.838**	**0.854**	**RDSH (adaptive)**	**0.801**	**0.832**	**0.842**	**0.851**
RDSH ($\beta = 0.5$)	0.600	0.812	0.820	0.854	RDSH ($\beta = 0.5$)	0.751	0.815	0.826	0.851

4. CONCLUSION

In this paper, we propose a robust deep supervised hashing method called RDSH. RDSH uses CNN for feature learning and hash function learning simultaneously. The learning procedure of RDSH is supervised by a robust pairwise loss and a quantitation loss. The robust pairwise loss is used for similarity-preserving learning, which simultaneously makes the learning procedure more robust to noisy training data. The quantitation loss guides the CNN to output binary codes. Comprehensive experiments on CIFAR-10 and NUS-WIDE datasets show the robustness and effectiveness of RDSH.

ACKNOWLEDGEMENTS

This work was supported in part by NSFC-Shenzhen Robot Jointed Founding (U1613215), in part by Shenzhen Municipal Development and Reform Commission (Disciplinary Development Program for Data Science and Intelligent Computing), and in part by Key-Area Research and Development Program of Guangdong Province (#2019B010137001).

REFERENCES

[1] Andoni, A., "Nearest Neighbor Search in High-Dimensional Spaces," International Symposium on Mathematical Foundations of Computer Science, 1-1(2011).
[2] Indyk, P. and Motwani, R., "Approximate nearest neighbors: towards removing the curse of dimensionality," Proceedings of the thirtieth annual ACM symposium on Theory of computing, 604-613(1998).
[3] Ji, J., Li, J., Yan, S., Zhang, B., and Tian, Q., "Super-bit locality-sensitive hashing," Proceedings of the 25th International Conference on Neural Information Processing Systems, 108-116(2012).
[4] Charikar, M. S., "Similarity estimation techniques from rounding algorithms," Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, 380-388(2002).
[5] Weiss, Y., Torralba, A., and Fergus, R., "Spectral hashing," Advances in neural information processing systems, 1753-1760(2008).
[6] Gong, Y. and Lazebnik, S., "Iterative quantization: A procrustean approach to learning binary codes," 2011 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2011, 817-824(2011).
[7] Kong, W. and Li, W.-J., "Double-bit quantization for hashing," Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, 634-640(2012).
[8] Liu, W., Wang, J., Kumar, S., and Chang, S.-F., "Hashing with graphs," Proceedings of the 28th International Conference on International Conference on Machine Learning, 1-8(2011).
[9] Kulis, B. and Darrell, T., "Learning to hash with binary reconstructive embeddings," Advances in neural information processing systems, 1042-1050(2009).
[10] Liu, W., Wang, J., Ji, R., Jiang, Y.-G., and Chang, S.-F., "Supervised hashing with kernels," IEEE Conference on Computer Vision and Pattern Recognition, 2074-2081(2012).
[11] Lin, R.-S., Ross, D. A., and Yagnik, J., "Spec hashing: Similarity preserving algorithm for entropy-based coding," 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 848-854(2010).
[12] Lin, G., Shen, C., Shi, Q., Van den Hengel, A., and Suter, D., "Fast supervised hashing with decision trees for high-dimensional data," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 1963-1970(2014).
[13] Shen, F., Shen, C., Liu, W., and Tao Shen, H., "Supervised discrete hashing," Proceedings of the IEEE conference on computer vision and pattern recognition, 37-45(2015).
[14] Kang, W.-C., Li, W.-J., and Zhou, Z.-H., "Column sampling based discrete supervised hashing," Thirtieth AAAI conference on artificial intelligence2016).
[15] Xia, R., Pan, Y., Lai, H., Liu, C., and Yan, S., "Supervised hashing for image retrieval via image representation learning," Twenty-eighth AAAI conference on artificial intelligence2014).
[16] Lai, H., Pan, Y., Liu, Y., and Yan, S., "Simultaneous feature learning and hash coding with deep neural networks," Proceedings of the IEEE conference on computer vision and pattern recognition, 3270-3278(2015).
[17] Li, W.-J., Wang, S., and Kang, W.-C., "Feature learning based deep supervised hashing with pairwise labels," Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, 1711-1717(2016).

[18] Zhu, H., Long, M., Wang, J., and Cao, Y., "Deep Hashing Network for efficient similarity retrieval," Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, 2415-2421(2016).
[19] Li, Q., Sun, Z., He, R., and Tan, T., "Deep supervised discrete hashing," Proceedings of the 31st International Conference on Neural Information Processing Systems, 2479-2488(2017).
[20] Cakir, F., He, K., Bargal, S. A., and Sclaroff, S., "Hashing with mutual information," IEEE transactions on pattern analysis machine intelligence 41(10), 2424 - 2437 (2019).
[21] He, K., Cakir, F., Adel Bargal, S., and Sclaroff, S., "Hashing as tie-aware learning to rank," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4023-4032(2018).
[22] Chatfield, K., Simonyan, K., Vedaldi, A., and Zisserman, A., "Return of the Devil in the Details: Delving Deep into Convolutional Nets," Proceedings of the British Machine Vision Conference2014).
[23] Krizhevsky, A., Sutskever, I., and Hinton, G. E., "ImageNet classification with deep convolutional neural networks," Proceedings of the 25th International Conference on Neural Information Processing Systems, 1097-1105(2012).
[24] He, K., Zhang, X., Ren, S., and Sun, J., "Deep residual learning for image recognition," Proceedings of the IEEE conference on computer vision and pattern recognition, 770-778(2016).
[25] Ding, N. and Vishwanathan, S., "t-Logistic regression," Advances in Neural Information Processing Systems, 514-522(2010).
[26] Krizhevsky, A. and Hinton, G., "Learning multiple layers of features from tiny images," Citeseer2009.
[27] Chua, T.-S., Tang, J., Hong, R., Li, H., Luo, Z., and Zheng, Y., "NUS-WIDE: a real-world web image database from National University of Singapore," Proceedings of the ACM international conference on image and video retrieval, 48(2009).
[28] Vedaldi, A. and Lenc, K., "Matconvnet: Convolutional neural networks for matlab," Proceedings of the 23rd ACM international conference on Multimedia, 689-692(2015).