symmetry

MDPI

Article

Unsupervised Hashing with Gradient Attention

Shaochen Jiang [1], Liejun Wang [2,*], Shuli Cheng [2], Anyu Du [2] and Yongming Li [2]

[1] College of Software, Xinjiang University, Urumqi 830046, China; jsc@stu.xju.edu.cn
[2] College of Information Science and Engineering, Xinjiang University, Urumqi 830046, China; slcaydxju@stu.xju.edu.cn (S.C.); anydxju@xju.edu.cn (A.D.); lym@xju.edu.cn (Y.L.)
* Correspondence: wljxju@xju.edu.cn; Tel.: +86-139-9981-6618

Received: 2 July 2020; Accepted: 14 July 2020; Published: 18 July 2020

Abstract: The existing learning-based unsupervised hashing method usually uses a pre-trained network to extract features, and then uses the extracted feature vectors to construct a similarity matrix which guides the generation of hash codes through gradient descent. Existing research shows that the algorithm based on gradient descent will cause the hash codes of the paired images to be updated toward each other's position during the training process. For unsupervised training, this situation will cause large fluctuations in the hash code during training and limit the learning efficiency of the hash code. In this paper, we propose a method named Deep Unsupervised Hashing with Gradient Attention (UHGA) to solve this problem. UHGA mainly includes the following contents: (1) use pre-trained network models to extract image features; (2) calculate the cosine distance of the corresponding features of the pair of images, and construct a similarity matrix through the cosine distance to guide the generation of hash codes; (3) a gradient attention mechanism is added during the training of the hash code to pay attention to the gradient. Experiments on two existing public datasets show that our proposed method can obtain more discriminating hash codes.

Keywords: deep learning; deep hash; image retrieval; pairwise label; unsupervised hashing

1. Introduction

Image retrieval is a hotspot in multimedia data retrieval, and there are many different methods today. The content-based image retrieval method [1] cannot meet the requirements of timeliness and retrieval precision due to its own limitations. In order to obtain higher retrieval accuracy, early researchers preferred the nearest neighbor search (NN). NN tends to retrieve the closest object from the dataset, and the complexity increases linearly with the amount of data, so its time performance cannot be guaranteed. Nowadays, there are a lot of high-dimensional multimedia data on social networks, the existing NN-based retrieval methods cannot obtain ideal retrieval results and an acceptable retrieval time. Therefore, researchers begin to pay attention to approximate nearest neighbor (ANN) [2–9]. Compared with the traditional retrieval method [1], another effective solution is the hashing method [2], through which high-dimensional media data can be converted into hash codes in the binary space.

In recent years, hashing has become a popular sub-region in ANN search [6,10,11]. With low storage cost and high query speed, the hash method has become the preferred method for effective ANN search. Hashing methods include two subcategories: (1) data-dependent hashing; (2) data-independent hashing. The process for data-independent hashing to obtain the hash code depends on the random mapping, and a reliable hash function cannot be obtained. Liu et al. [3] uses a set of hash functions to create multiple hash tables. After hash mapping, similar data points are more likely to collide in these hash tables, but the probability of collision of different points is very small. Therefore, similar images and non-similar images can be well distinguished. On this basis, Bai et al. [12] proposed to transfer the

distance constraint of the hash code from the Hamming space to the Euclidean space. Data-dependent hashing methods use image colors, textures, shapes, or other features to directly generate hash codes and construct hash functions. Current researchers are committed to using deep learning methods to map features extracted from Convolutional Neural Network (CNN) to Hamming space through a hash function.

The data-dependent hashing method includes two subclasses: supervised hashing and unsupervised hashing. The difference between the two subclasses is whether the label information is used to construct the hash function. Gong et al. [13] was a representative algorithm in unsupervised hashing. The application of the Principal Components Analysis (PCA) algorithm could effectively reduce the dimensions of the original dataset, and at the same time constructed a binary hypercube. Each vertex of the hypercube represented the data point in the dataset, the quantization error was minimized by the rotation matrix. In supervised hashing methods [14–19], the construction process of the hash function in these methods used similar information of the paired images. In binary space, the Hamming distance of the image pair was determined by the image pair similarity.

The further development of CNN shows an excellent performance in feature extraction [20]. Existing supervised deep hashing methods have achieved a good performance [21–26]. However, for the massive image data existing in the network, it is not realistic to manually label them, and the hash method that relies on annotated data may not be suitable for practical applications.

In order to use these massive amounts of unlabeled data, the researchers proposed deep unsupervised hashing method. The learning process of the hash code in the early deep unsupervised hashing method was completed by the deep autoencoder [27,28]. However, the variability of natural images in terms of position, color, posture, etc., affected the representativeness of the learned hash code. Erin Liong et al. [29] maximized the representation of hash codes. Lin et al. [30] kept the hash code of the rotating image unchanged to learn hash codes. Jin et al. [31] further studied the rotation invariance between different images in the same category label. Yang et al. [32] used two half-Gaussian distributions to estimate the cosine distance distribution of different data points. Based on this hypothetical estimate, a pair-wise loss function was designed to retain semantic information. Generative adversarial networks (GAN) were widely used in various fields [33], Ghasedi Dizaji et al. [34] used a shared parameter generator and discriminator to optimize the hash function in the adversarial. On the basis of GAN, Deng et al. [35] constructed a semantic similarity matrix by using feature similarity and nearest neighbor similarity to guide the construction of hash codes.

The optimization algorithm commonly used in current deep learning models is gradient descent. Huang et al. [36] proposed that the hash model based on gradient descent optimization may fall into a dilemma during the optimization process, the hash codes of the paired images will always be updated in the direction of the other side, and the worst case is that the paired hash codes will exchange directions after being updated. This phenomenon is the "dilemma" of gradient descent. After falling into this dilemma, the Hamming distance corresponding to hash code will not be reduced, affecting the learning process, since this problem is caused by gradient changes in the process of updating hash codes in backpropagation. Selectively ignoring or reducing the gradient of one of the hash codes during backpropagation is an effective way to solve this problem. The gradient attention mechanism is designed to focus on the gradient of the paired image hash codes in supervised hashing.

Existing unsupervised deep hashing methods usually use a pre-trained network model to construct a similarity matrix to guide the generation of hash codes. Therefore, the above problems still exist in unsupervised learning, through the experimental results show that after adding gradient attention mechanism, unsupervised hash method achieved better performance.

2. UHGA Mmethod

2.1. Deep Hash Model

For unsupervised hashing, $\boldsymbol{X} = \{x_i\}_{i=1}^{N}$ represents a dataset containing N images, where x_i is the i-th image. Unsupervised hashing aims to obtain the binary code for each data point in the dataset

and construct a hash function $M(\theta; x)$ to generate this binary code, where θ represents the parameters of the deep model. Finally, after each image x_i passes through the deep model M, it will be mapped into a k-dimensional vector $\boldsymbol{Z} = \{z_i\}_{i=1}^{n}$, where z_i the i-th k-dimensional vector. After performing a $b_i = sign(z_i)$ binarization operation on this k-dimensional vector, we obtain its corresponding hash code $\left\{b_i \in \{\pm 1\}^k\right\}_{i=1}^{n}$, where b_i the i-th hash code. The sign function is expressed as:

$$sign(x) = \begin{cases} 1 \; if \; x \geq 0, \\ -1 \; if \; x < 0 \end{cases} \tag{1}$$

Unsupervised hashing usually guides the learning of hashing codes by constructing the semantic similarity matrix between image pairs. In this paper, we extract the features of FC-7 layer from n randomly selected data points in the dataset through the pre-trained Alexnet network to construct the similarity matrix. Most of the current deep hash algorithms use VGG-F network in MATLAB as the benchmark. In this paper, our experiment is implemented on the deep learning framework Pytorch. To ensure the fairness of the experiment, Alexnet, which has the same structure as VGG-F network, is selected as the benchmark network. For the input N data points, the corresponding feature matrix is obtained after the pre-trained Alexnet. According to the feature matrix, calculating the cosine distance of the corresponding feature of any two data points, and the cosine distance matrix D is constructed.

$$D = \frac{F_i^T F_j}{\|F_i\| \times \|F_j\|} \tag{2}$$

where F_i represents the feature of FC-7 layer.

Then, the similarity matrix S is constructed as follows through the cosine distance matrix:

$$S_{ij} = \begin{cases} 1, & if \; d(i,j) < D.mean - \alpha, \\ 0, & others, \\ -1, & if \; d(i,j) > D.mean + \beta \end{cases} \tag{3}$$

where $D.mean$ represents the average value of cosine distance matrix D, α and β are threshold parameters,$S_{ij} = 1$ means that x_i is semantically similar to x_j, $S_{ij} = -1$ means the opposite, and $S_{ij} = 0$ means that the semantic relationship between x_i and x_j is uncertain.

In the training process, the *sign* function cannot perform gradient training, *tanh* is generally used instead to relax each binary code into a continuous hash code $h \in [-1, +1]^k$, where h^k and b^k denote the k-th bit of h and b.

To preserve the similarity of the paired image in the hash code, we use the inner product of the hash code to construct prediction similarity matrix $\mathfrak{I}$:

$$\mathfrak{I}_{ij} = h_i^T h_j \tag{4}$$

where h_i and h_j are relaxed hash codes.

To maintain the semantic information between image pairs, we minimize the L2 distance $\left(\mathfrak{I}_{ij} - S_{ij}\right)^2$ between the similarity matrix constructed by the feature matrix and the predicted similarity matrix. The resulting objective function is:

$$\min_{\theta} l(\theta) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \left|S_{ij}\right| \left(\mathfrak{I}_{ij} - S_{ij}\right)^2 \tag{5}$$

where $\mathfrak{I}_{ij} = \frac{1}{k} h_i^T h_j$, $h_i = tanh(M(\theta; x))$ and θ is the parameter of the network model.

The commonly used method is to update the parameter θ by calculating the gradient of the loss function. First we define $v_i = M(\theta; x)$, the derivative of v_i to θ can be directly calculated, and then calculate the derivative of $l(\theta)$ to v_i by Equation(6).

$$\frac{\partial l(\theta)}{\partial v_i} = \frac{2}{kn^2}\sum_{i=1}^{n}\left(1-h_i^2\right)\odot\left(\sum_{j=1}^{n}\left|S_{ij}\right|\left(\frac{1}{k}h_i^T h_j - S_{ij}\right)h_j + \left(\frac{1}{k}h_i^T h_j - 1\right)h_i\right) \tag{6}$$

Gradient descent is usually used to optimize parameters $\theta \leftarrow \theta - \mu\frac{\partial l}{\partial \theta}$. In the optimization process, the backpropagation method is used to calculate the gradient of the model parameter θ, the partial derivative $\partial l/\partial \Im$ is first calculated, and then the partial derivative h_i^k can be calculated by Equation (7).

$$\frac{\partial l}{\partial h_i^k} = \sum_j \frac{\partial l}{\partial \Im_{ij}}\frac{\partial \Im_{ij}}{\partial h_i^k} \tag{7}$$

where h_i^k denote the k-th bit of h_i.

Finally, the gradient $\partial l/\partial \theta$ can be calculated by (8):

$$\frac{\partial l}{\partial \theta} = \sum_i \frac{\partial l}{\partial h_i}\frac{\partial h_i}{\partial \theta} \tag{8}$$

where $\frac{\partial l}{\partial h_i} = \left[\frac{\partial l}{\partial h_i^1}, \frac{\partial l}{\partial h_i^2}, \ldots, \frac{\partial l}{\partial h_i^k}\right]^T$.

2.2. The Dilemma of Gradient Decline

The purpose of network model optimization is to update the parameter θ of the model in the training process, but after updating θ, the detailed changes in h cannot be displayed directly, in order to analyze the change process of h, [36] proposes a lemma to analyze the update behavior of h.

Lemma 1: *for a given composite function $g(y)$, where x is a vector and $y = f(x)$ is a scaling function.*

Suppose x is updated by gradient descent, that is $x^+ = x - \alpha\frac{\partial g}{\partial x}$, where α is the step size.
Conclusion: $y^+ = f(x^+) \approx \mathrm{y} - \hat{\mathrm{a}}\frac{\partial g}{\partial y}$, where â is some positive scaler. i.e., y update along negative gradient.

Proof: Calculate the first-order Taylor expansion of $f(x^+)$ on x:

$$\begin{aligned} y^+ - \mathrm{y} &= f(x^+) - f(x) \approx \left\langle \frac{\partial y}{\partial x}, x^+ - x\right\rangle = -\alpha\left\langle \frac{\partial y}{\partial x}, \frac{\partial g}{\partial x}\right\rangle \\ &= -\alpha\left\langle \frac{\partial y}{\partial x}, \frac{\partial g}{\partial y}, \frac{\partial y}{\partial x}\right\rangle = -\alpha\left\|\frac{\partial y}{\partial x}\right\|^2\frac{\partial g}{\partial y} = -\hat{\mathrm{a}}\frac{\partial g}{\partial y} \end{aligned} \tag{9}$$

where $\hat{\mathrm{a}} = \alpha\left\|\frac{\partial y}{\partial x}\right\|^2$. This proof is completed. □

By substituting $x = \theta$, $y = h$ and $f(x) = tanh(M(\theta))$ into Equation (9), it can be seen that when θ is updated by gradient descent, h changes along its negative gradient direction, and then the change in h can be observed by gradient.

For a pair of input images $\left(x_i, x_j\right)$, assume x_i is similar to x_j, i.e., $s_{ij} = 1$. If one of the bits in the current corresponding continuous hash code are $h_i = 1$ and $h_j = -1$ respectively, then the loss can be defined as $\ell = 1 - \Im_{ij} = 1 - h_i^T h_j$. At this time, $\ell > 0$, the current hash code needs to be updated. The gradients of h_i and h_j are:

$$\frac{\partial l}{\partial h_i} = \frac{\partial l}{\partial \Im_{ij}}\frac{\partial \Im_{ij}}{\partial h_i} = -h_j \tag{10}$$

$$\frac{\partial l}{\partial h_j} = \frac{\partial l}{\partial \Im_{ij}} \frac{\partial \Im_{ij}}{\partial h_j} = -h_i \tag{11}$$

Based on the lemma above, the updated values of h_i and h_j are expressed as h_i^+ and h_j^+ respectively, which can be expressed as:

$$h_i^+ = h_i - \alpha_{ij} \frac{\partial l}{\partial h_i} = h_i + \alpha_{ij} h_j \tag{12}$$

$$h_j^+ = h_j - \alpha_{ji} \frac{\partial l}{\partial h_j} = h_j + \alpha_{ji} h_i \tag{13}$$

where α_{ij} and α_{ji} are some positive scalers.

It can be seen from Equation (12) and (13) that, in the process of optimization, h_i moves to h_j direction and tends to have the same value as h_j. The update of h_j is the same as h_i. Note that when $h_i = h_j$, $\Im_{ij} = S_{ij}$, the hash codes are optimal. However, in some cases i.e., $h_i = -h_j = 1$, $\Im_{ij} = S_{ij}$, and $h_i^T h_j = -1$ when $\alpha_{ij} = \alpha_{ji} = 2$, $h_i^+ = h_j = 1$ and $h_j^+ = h_i = -1$, their inner product is still $h_i^{+T} h_j^+ = -1$ after updating, therefore, the updated loss remains unchanged. In the next round of update, h_i and h_j will switch their symbols, and their corresponding losses will not be reduced, which will happen repeatedly. This dilemma also occurs when two different pairs of code are updated to stay away from each other.

Updating one of the paired hash codes during backpropagation is an effective solution, when their similarity prediction is not correct. When the prediction is correct, h_i and h_j are updated towards their sign value, which can effectively avoid the problem of symbol switching. When updating h_i and h_j the weight of the gradient should be carefully selected so as to update only one code when the similarity of prediction image is not correct, and two codes when the similarity of prediction is correct.

2.3. Gradient Attention Network

In this paper, we use gradient attention network to focus on the gradients of h_i^k and h_j^k, and generate weights for the derivatives of h_i^k and h_j^k before backpropagation to θ. We use the weighted derivatives of h_i^k and h_j^k instead of the original derivatives to calculate the gradients of θ. For all training pairs, the weighted derivatives of h_i^k and h_j^k are calculated as follows:

$$\frac{\partial l}{\partial h_i^k} = \sum_j \xi_{ij,i}^k \frac{\partial l}{\partial \Im_{ij}} \frac{\partial \Im_{ij}}{\partial h_i^k} \tag{14}$$

$$\frac{\partial l}{\partial h_j^k} = \sum_i \xi_{ij,j}^k \frac{\partial l}{\partial \Im_{ij}} \frac{\partial \Im_{ij}}{\partial h_j^k} \tag{15}$$

where $\xi_{ij,i}^k$ and $\xi_{ij,j}^k$ are the weights generated by the derivative of a pair of binary bits h_i^k and h_j^k respectively. A new gradient $g(\varphi)$ can be obtained by substituting (14), (15) into (8). The parameters of the gradient attention network are expressed by φ.

For the hash model optimized by gradient descent, the change in the value of the corresponding hash code h_i^k of the paired image $\left(x_i, x_j\right)$ is updated by the derivative related to the loss based on the original value during the optimization process. For the gradient attention network, all the inputs are h_i^k, $\frac{\partial l}{\partial \Im_{ij}} \frac{\partial \Im_{ij}}{\partial h_i^k}$ and $\frac{\partial l}{\partial h_i^k}$.

The gradient attention network first generates the feature value $y_{ij,i}^k$ for the k-th bit of the hash code corresponding to the paired image, then calculates the weight of the k-th bit and normalizes it by the softmax function. The calculation process is as follows:

$$\xi_{ij,i}^k = \frac{exp\left(y_{ij,i}^k\right)}{\exp\left(y_{ij,i}^k\right) + exp\left(y_{ij,j}^k\right)} \tag{16}$$

$$\xi_{ij,j}^k = \frac{\exp\left(y_{ij,j}^k\right)}{\exp\left(y_{ij,i}^k\right) + \exp\left(y_{ij,j}^k\right)} \tag{17}$$

Figure 1 shows the deep unsupervised hash network model. The gradient attention network consists of two fully connection layers, each of which contains 100 hidden units.

The detailed learning process of the UHGA model is shown in Algorithm 1.

Algorithm 1. UHGA

Input:

Training set $\boldsymbol{X} = \{x_i\}_{i=1}^n$;
Code length k;

Output:

Updated network parameters θ and hash codes B.

Initialization:

Initialize the Alexnet;
Initialize parameters θ from the pre-trained Alexnet.
Extracting features of Alexnet FC-7 layer.
Using (2) to calculate cosine distance of paired image features.
Using (3) to construct similarity matrix S.

Repeat

a mini batch of images from $\boldsymbol{X} = \{x_i\}_{i=1}^n$, and for each image x_i, follow each step below:
•Calculate $l(\theta)$ and obtain the derivative $\partial l / \partial \Im$ in the process of backpropagation;
•Using h_i^k, $\frac{\partial l}{\partial \Im_{ij}} \frac{\partial \Im_{ij}}{\partial h_i^k}$ and $\frac{\partial l}{\partial h_i^k}$ as input of gradient attention network to calculate attention weights $y_{ij,i}^k$;
•Calculate $\xi_{ij,i}^k$ and $\xi_{ij,j}^k$ by (16) and (17);
•Calculate the new derivatives $\frac{\partial l}{\partial h_i^k} = \sum_j \xi_{ij,i}^k \frac{\partial l}{\partial \Im_{ij}} \frac{\partial \Im_{ij}}{\partial h_i^k}$ and obtain gradient of θ as $g(\varphi)$ in the process of backpropagation;
•Update θ by (18);
•Calculate the loss $l(\theta - \alpha g(\varphi))$ after update;
•Update φ;
• $\theta \leftarrow \theta^+$;

Until number of iterations reached

2.4. Optimization

The gradient attention network can solve the problem of the exchanging positions of consecutive hash codes corresponding to pairs of images after updating. Due to this problem, the loss after updating remains unchanged, as the performance of attention weight can be evaluated by the loss reduced during the updating process. In gradient attention networks, the new gradient is first generated and then the parameter θ of the hash model is updated by Equation (18).

$$\theta^+ = \theta - \alpha g(\varphi) \tag{18}$$

The updated loss is $l(\theta^+) = l(\theta - \alpha g(\varphi))$, and the reduced loss is $l(\theta) - l(\theta^+)$. After parameter θ is updated, the goal is still to minimize the loss; the goal for optimizing gradient attention is:

$$\min_{\varphi} l(\theta - \alpha g(\varphi)) \tag{19}$$

For Equation (19), θ can be optimized using Stochastic Gradient Descent (SGD) from Equation (6) and, as long as θ^+ is a function of $g(\varphi)$, φ can be optimized by any optimizer.

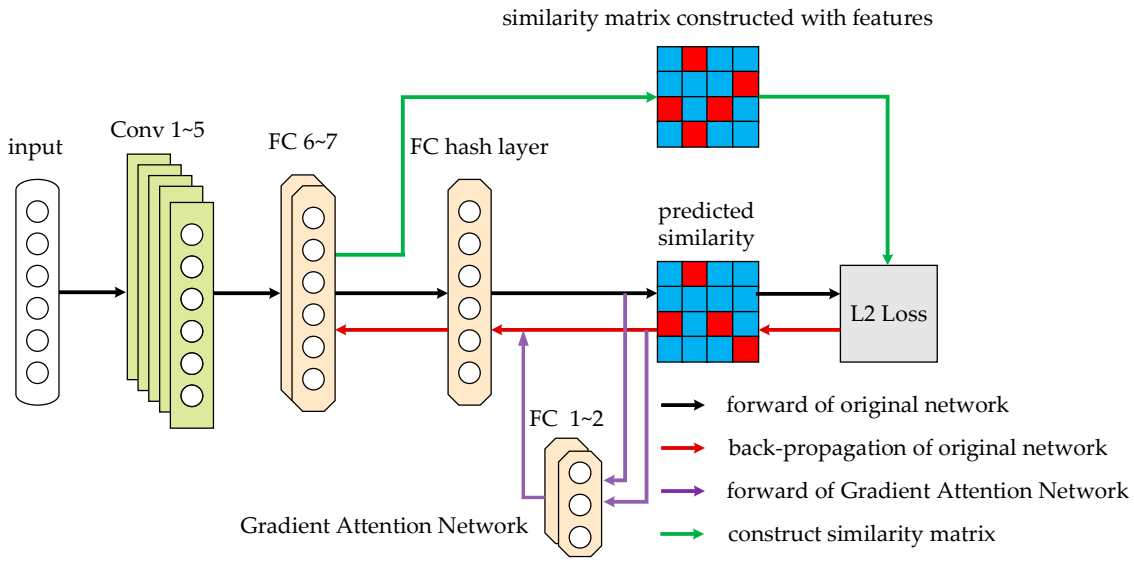

Figure 1. Deep Unsupervised Hashing with Gradient Attention (UHGA) network structure.

3. Experimental Results and Analysis

The UHGA model is based on the Alexnet implementation in Pytorch. Detailed configuration information for the experimental machine is shown in Table 1.

Table 1. Configuration information.

Item	Configuration
OS	Ubuntu 16.04 (X64)
GPU	4*GTX 1080ti

3.1. Datasets, Evaluation Metrics and Benchmarks

There are two authoritative image datasets: CIFAR-10 and NUS-WIDE datasets. The experiments have used these two datasets as benchmark datasets to compare with other methods in order to verify the feasibility of the UHGA model.

(1) CIFAR-10. A total of 60,000 images, including 10 categories, and each category contains 6000 images, which is a single-label dataset.

(1) NUS-WIDE. A total of 269,648 images, including 21 categories, each category is associated with at least 5000 images and each image belongs to one or more categories, which is a multi-label dataset.

In this article, the methods of SH (Spectral hashing) [37], SPH (Spherical hashing) [38], ITQ (Iterative quantization) [13], DSH (Density Sensitive Hashing) [39], SSDH (Semantic structure-based unsupervised deep hashing) [32], HashGAN (Unsupervised Deep Generative Adversarial Hashing Network) [34], and USDH (Unsupervised Semantic Deep Hashing) [31] are selected for comparison. We use Mean Average Precision (MAP), Precision–Recall curves (PR) and the precision curves of the first 1000 retrieval results (P@N) as evaluation metrics.

3.2. Hyperparameter Analysis

We use Equation (3) to construct a similarity matrix between images. There are two hyperparameters α and β in Equation (3). In order to analyze the influence of hyperparameters, we simply set the two hyperparameters to $\alpha = \eta(D.mean - D.min)$ and $\beta = \eta(D.max - D.mean)$. Then, we only need to adjust the hyperparameter η to observe the influence of the constructed similarity matrix on the retrieval accuracy. Table 2 shows the effect of different η on two datasets under different bits. In order to observe the change in MAP, we visualize it in Figure 2. Figure 2 shows that the value of MAP is relatively smooth when η is between 0.2-0.4, and when η continues to increase, MAP shows the opposite trend. Therefore, in subsequent experiments, the hyperparameter η is set to 0.3.

Table 2. Mean Average Precision (MAP) of different η.

η	CIFAR-10 (MAP@ALL) (bits)				NUS-WIDE (MAP@5000) (bits)			
	16	**32**	**64**	**128**	**16**	**32**	**64**	**128**
0	0.1015	0.3091	0.3744	0.3739	0.6571	0.6976	0.7230	0.7409
0.1	0.1015	0.3080	0.3413	0.3668	0.6546	0.6985	0.7198	0.7356
0.2	0.2940	0.3204	0.3464	0.3704	0.6480	0.7007	0.7183	0.7340
0.3	0.2841	0.3283	0.3508	0.3688	0.6444	0.6943	0.7213	0.7343
0.4	0.2854	0.3309	0.3470	0.3653	0.6434	0.6892	0.7179	0.7318
0.5	0.2837	0.3195	0.3404	0.3629	0.6235	0.6768	0.7131	0.6979
0.6	0.2822	0.3096	0.3276	0.3347	0.6199	0.6705	0.6940	0.6892
0.7	0.2199	0.2428	0.2650	0.2993	0.5761	0.6294	0.6661	0.7120
0.8	0.1677	0.2008	0.2380	0.2900	0.5239	0.5795	0.6282	0.6979
0.9	0.1576	0.1932	0.2334	0.2888	0.4920	0.5522	0.6168	0.6926

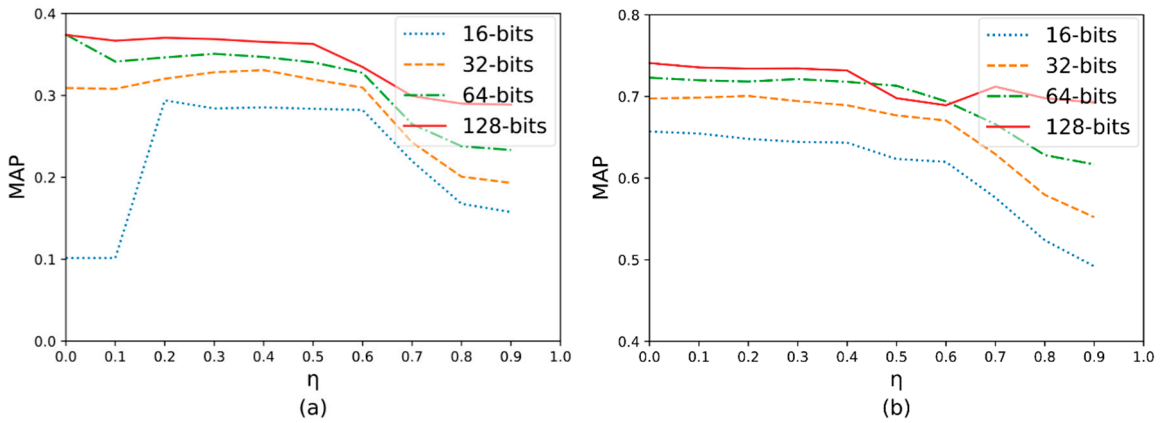

Figure 2. MAP on different η. (**a**) MAP on CIFAR-10; (**b**) MAP on NUS-WIDE.

3.3. The effect of Gradient Attention

In order to study the effect of gradient attention, we added the gradient attention mechanism to the method based on pairwise loss. We used four methods in Table 3 as comparative experiments: SSDH, SSDH+GA, Ours, Ours+GA, where SSDH is its original method, SSDH+GA means adding gradient attention mechanism in SSDH, Ours means only using our own designed loss function to train the unsupervised hash model, and Ours+GA means adding a gradient attention mechanism to our own loss function.

Table 3. MAP for Different Number of Bits.

Method	CIFAR-10 (MAP @ALL) (bits)				NUS-WIDE (MAP @5000) (bits)			
	16	**32**	**64**	**128**	**16**	**32**	**64**	**128**
SSDH [32]	0.2172	0.2313	0.2405	0.2351	0.5860	0.5788	0.6079	0.6127
SSDH+GA	0.2827	0.3018	0.3219	0.3462	0.6528	0.6637	0.6787	0.7032
Ours	0.2704	0.3133	0.3240	0.3452	0.6512	0.6846	0.7039	0.7125
Ours+GA	0.2936	0.3233	0.3499	0.3661	0.6601	0.6981	0.7224	0.7359

It can be seen from Table (3) that, after adding the attention mechanism, the MAP of SSDH [32] is significantly improved. We obtained good results using our own designed loss function, and the map value continued to increase after using the attention mechanism. Figure 3 shows the decreasing trend of training loss before and after adding the gradient attention (GA) mechanism in SSDH [32] and our method. It can be clearly seen that the loss decreases faster after adding the gradient attention mechanism, which helps to obtain a better hash code.

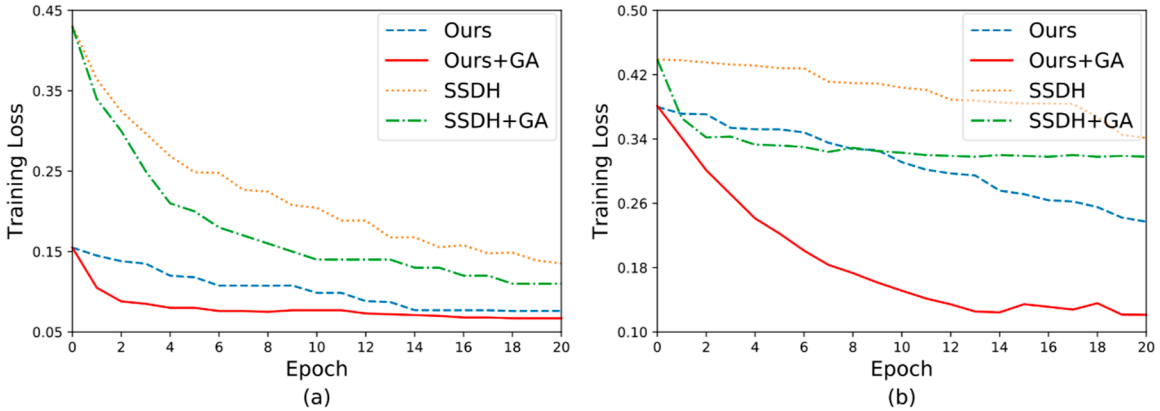

Figure 3. (**a**) Training loss on CIFAR-10; (**b**) training loss on NUS-WIDE.

The loss function we proposed in Table 3 has been greatly improved in MAP. This is also shown in the decreasing trend of training loss in Figure 3. The loss function we designed has more advantages in the generation of constrained hash codes. After adding the attention mechanism, our method and SSDH [31] are significantly improved on MAP. In Figure 3, the improvement in hash performance is reflected in the faster decline in training loss.

Figure 4 shows the effect of different learning rates on retrieval results. It can be seen from (a) and (b) in Figure 4 that, without the gradient attention mechanism, the retrieval quality will decline with the increase in the learning rate because, in the process of reverse propagation, when the learning rate is high, the network cannot converge. After using the gradient attention mechanism, the magnitude of the gradient descent is not only controlled by the learning rate, but also by the gradient attention. In the process of backpropagation, the gradient attention mechanism can select the appropriate gradient modification parameters for the network to obtain a better deep hash model.

The above situations indicate that the unsupervised method based on pairwise loss have a dilemma in gradient descent, and the design of the loss function also affects the probability of this dilemma. Obviously, the loss function we designed has a lower probability of dilemma. The addition of the gradient attention mechanism can effectively reduce the occurrence of this dilemma, thereby improving the performance of the hash method.

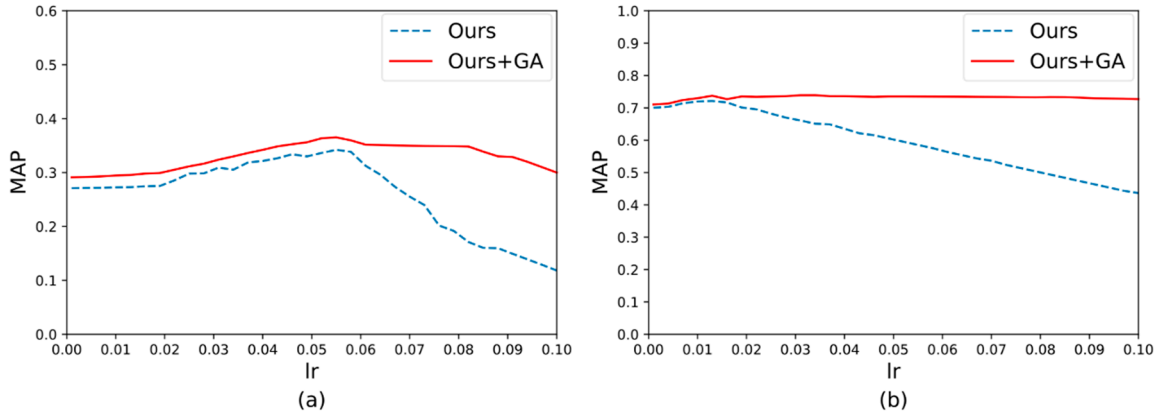

Figure 4. MAP on different learning rates (lr): (**a**) MAP on CIFAR-10; (**b**) MAP on NUS-WIDE.

3.4. Comparison of Experimental Results

In the training of the hash function, we select 500 images from each image category to form a training dataset, and select 100 images for each category to serve as the test dataset on CIFAR-10. All remaining image samples are used as retrieved data. We select 21 kinds of common images from the NUS-WIDE dataset. Each class is associated with at least 5000 images, with a total of 190,000 images. The NUS-WIDE test dataset consists of 100 images of each class and 2100 images in total. The training dataset contains 5000 randomly selected images, and the rest of the images are retrieved as the sample dataset.

Table 4 shows the comparison results of all comparative experiments on the evaluation metric MAP. We calculated the MAP value of the hash code length of all the comparison methods from 16 bits to 128 bits on the two datasets. In the measurement results of different hash code length, our method is 8.73%, 8.67%, 12.36%, 13.79% higher than USDH in the CIFAR-10 dataset, and 5.77%, 5.71%, 9.51%, 9.21% higher than USDH in the NUS-WIDE dataset.

Table 4. MAP for different numbers of bits.

Method	CIFAR-10 (MAP @ALL) (bits)				NUS-WIDE (MAP @5000) (bits)			
	16	32	64	128	16	32	64	128
SH [37]	0.1621	0.1575	0.1549	0.1548	0.4350	0.4129	0.4042	0.4100
SPH [38]	0.1664	0.1745	0.1931	0.1992	0.4458	0.4537	0.4926	0.5.00
ITQ [13]	0.1999	0.2098	0.2229	0.2312	0.5082	0.5139	0.5252	0.5311
DSH [39]	0.1686	0.1846	0.1955	0.2071	0.5123	0.5118	0.5110	0.5267
SSDH [32]	0.2172	0.2313	0.2405	0.2351	0.5860	0.5788	0.6079	0.6127
HashGAN [34]	0.2457	0.2678	0.3077	0.3422	0.4020	0.4207	0.4460	0.4883
USDH [31]	0.2613	0.3021	0.3122	0.3241	0.6407	0.6568	0.6587	0.6724
Ours+GA	0.2936	0.3233	0.3499	0.3661	0.6601	0.6981	0.7224	0.7359

Figure 5and Figure 6 show the PR and P@N curves on CIFAR-10, respectively. From Figure 5, the area under the PR curve corresponding to our method is the largest, and our method shows better results as the length of the hash code increases. As can be seen in Figure 6, as the number of retrieval results continues to increase, the retrieval precision of all methods shows a downward trend, but our method always shows the best results.

Figures 7 and 8 show the PR and P@N curve on NUS-WIDE respectively. Figure 7 shows that our method still has higher performance on multi-label datasets. Although the precision of our method is lower than USDH when the recall is between 0-0.1, the area under the PR curve corresponding to our method is still the largest overall; this also shows that our method is more stable. Figure 8 shows that the n curve is not as obvious as the downward trend in Figure 6, and our method still consistently shows the best performance in retrieval accuracy.

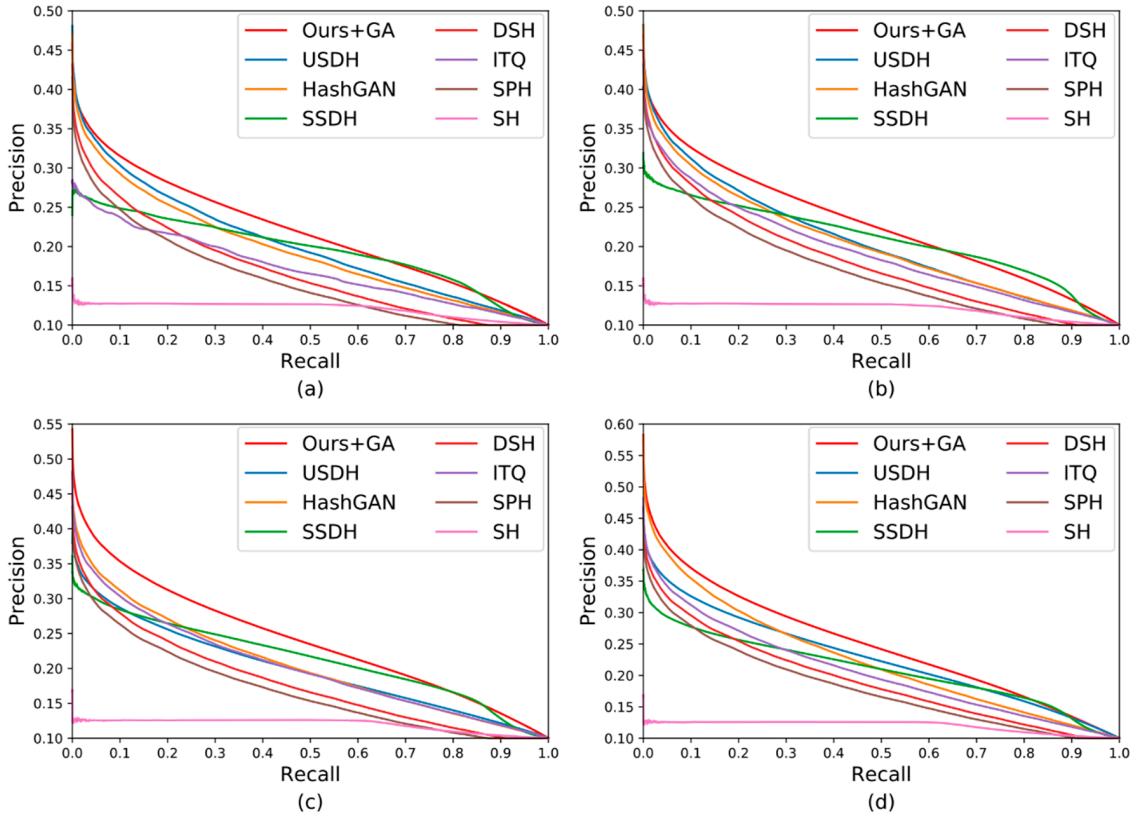

Figure 5. PR on CIFAR-10. (**a**) PR with code length 16; (**b**) PR with code length 32; (**c**) PR with code length 64; (**d**) PR with code length 128.

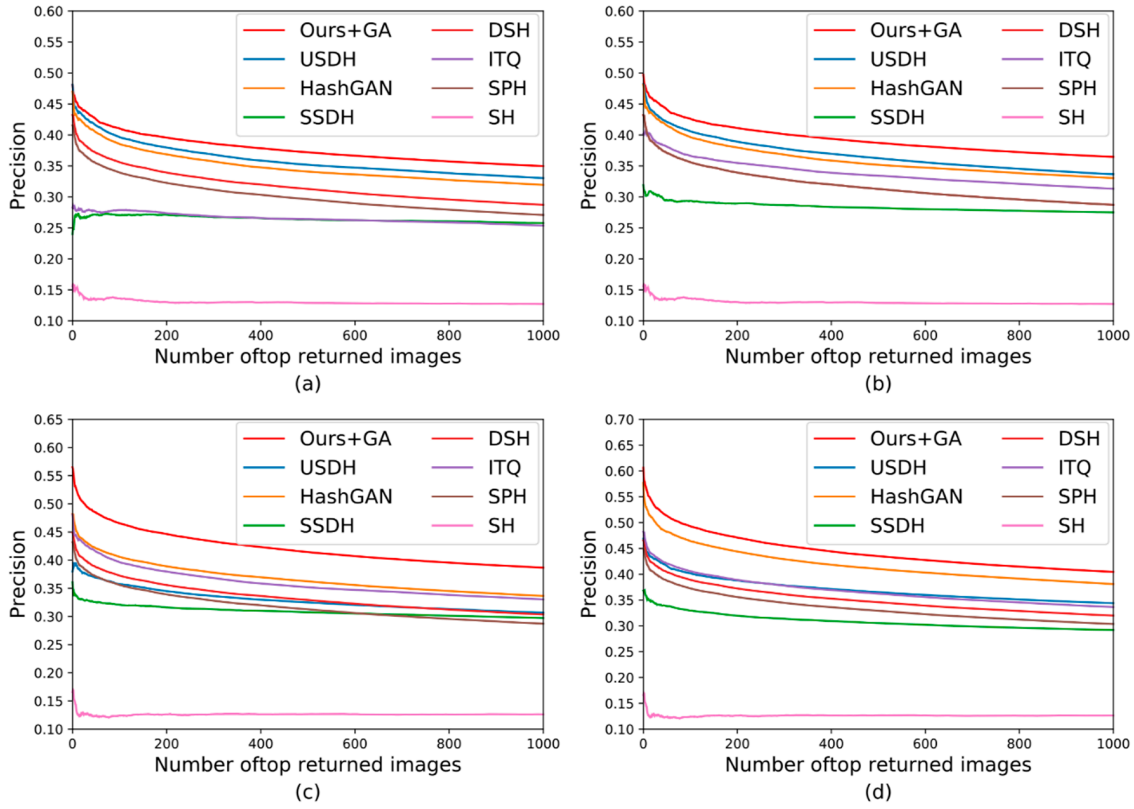

Figure 6. P@N on CIFAR-10. (**a**) P@N with code length 16; (**b**) P@N with code length 32; (**c**) P@N with code length 64; (**d**) P@N with code length 128.

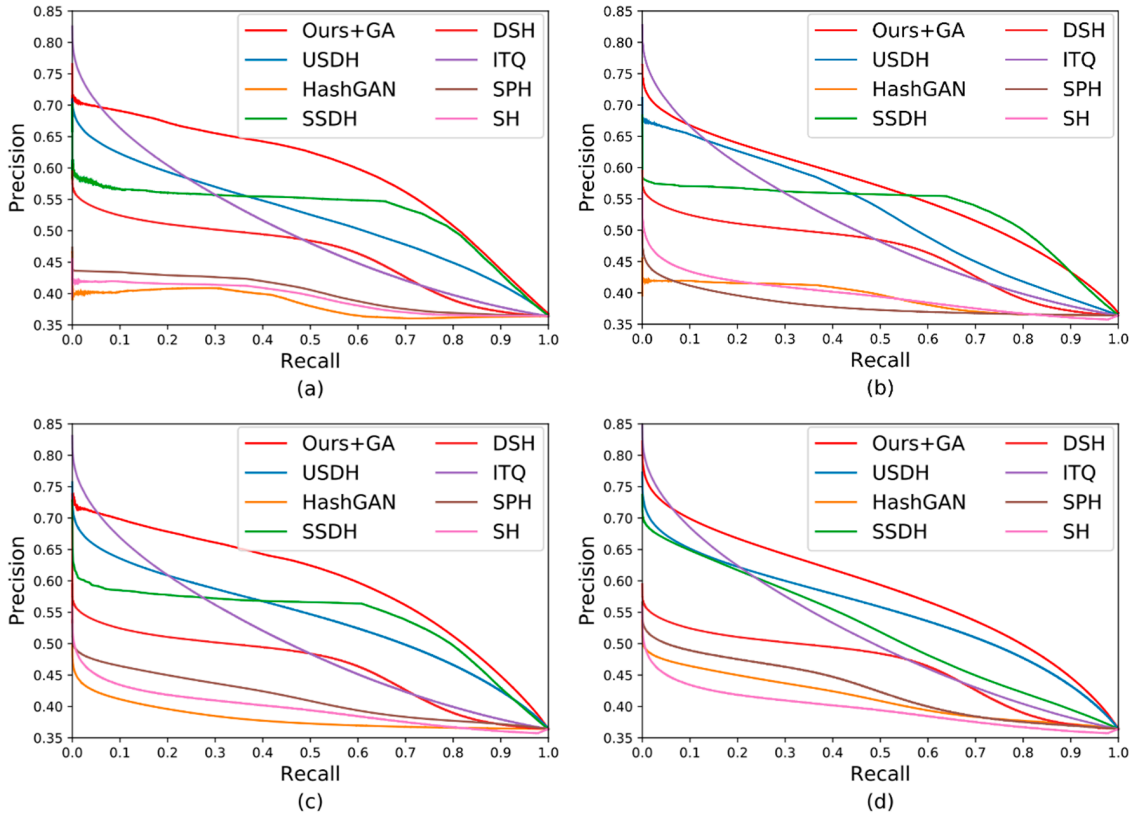

Figure 7. PR on NUS-WIDE. (**a**) PR with code length 16; (**b**) PR with code length 32; (**c**) PR with code length 64; (**d**) PR with code length 128.

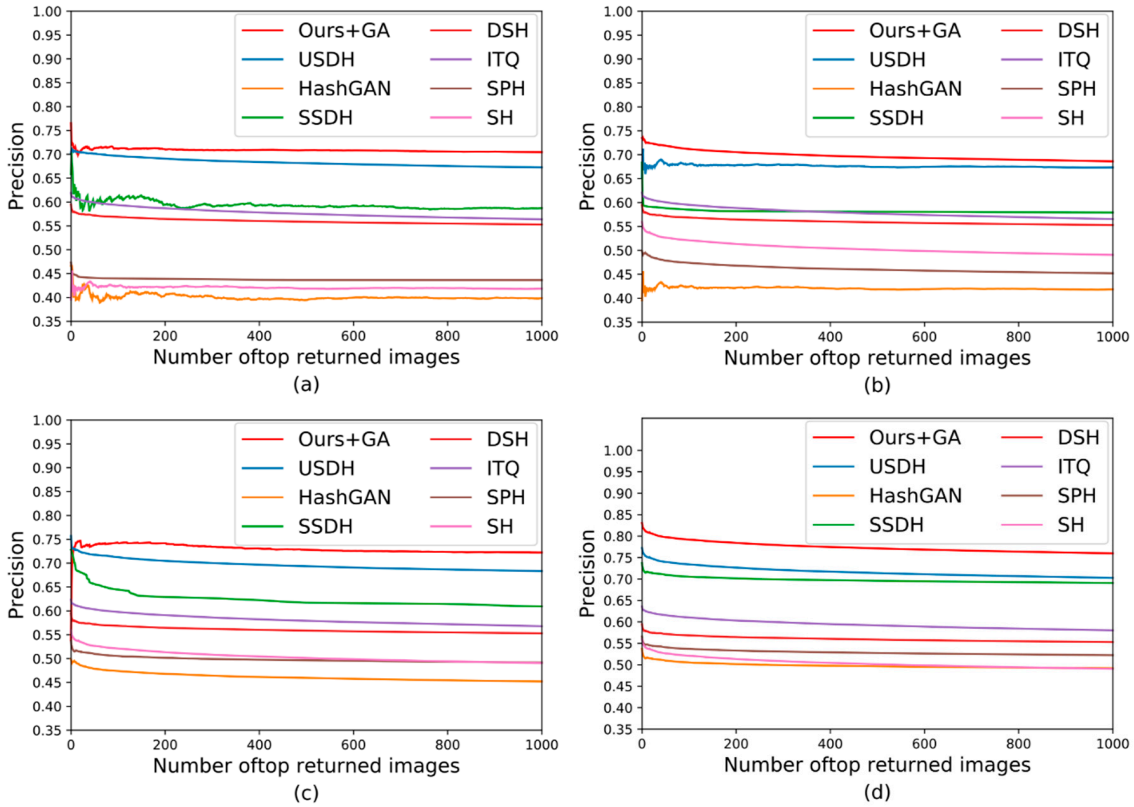

Figure 8. P@N on NUS-WIDE. (**a**) P@N with code length 16; (**b**) P@N with code length 32; (**c**) P@N with code length 64; (**d**) P@N with code length 128.

The above experimental results show that our unsupervised hash method achieves the best performance. Compared with other methods, the advantages of our proposed method are as follows: (1) the designed loss function can generate a hash code with a greater performance (2) by adding a gradient attention mechanism to reduce the impact of the dilemma in gradient descent during training.

4. Conclusions

In unsupervised deep hashing, a similarity matrix is usually constructed by extracting image features and used in the training of neural networks. Gradient descent is used to update parameters during training. However, for unsupervised hashing methods based on paired image training, there may be a problem in the hash code update process, as the hash codes corresponding to paired images are updated toward each other. After the network parameters are updated, the corresponding hash code changes its location, but the overall loss remains unchanged. This paper presents a Deep Unsupervised Hashing with Gradient Attention (UHGA) algorithm based on the deep learning end-to-end model. UHGA uses the cosine distance of features to construct similarity information between images, and the gradient attention mechanism in the process of gradient descent, which successfully obtains a better deep hash model and significantly optimizes the retrieval quality. Experiments on two datasets show that UHGA significantly improves the latest hash methods.

Author Contributions: Conceptualization, L.W. and S.J.; methodology, Y.L. and S.C.; software, S.J.; validation, A.D., Y.L. and L.W.; formal analysis, Y.L. and S.C.; writing—original draft preparation, S.J.; writing—review and editing, L.W.; visualization, A.D.; All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Xinjiang Uygur Autonomous Region Natural Science Foundation Project (2020D01C058), National Natural Science Foundation Project (61771416 and U1903213), Xinjiang Uygur Autonomous Region Higher Education Innovation Project (XJEDU2017T002), and Xinjiang Uygur Autonomous region graduate research and innovation project (XJ2019G039).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Liu, Y.; Zhang, D.; Lu, G.; Ma, W.Y. A survey of content-based image retrieval with high-level semantics. *Pattern Recognit.* **2007**, *40*, 262–282. [CrossRef]
2. Tang, J.; Li, Z.; Wang, M.; Zhao, R. Neighborhood discriminant hashing for large-scale image retrieval. *IEEE Trans. Image Process.* **2015**, *24*, 2827–2840. [CrossRef] [PubMed]
3. Paulevé, L.; Jégou, H.; Amsaleg, L. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern Recognit. Lett.* **2010**, *31*, 1348–1358. [CrossRef]
4. Andoni, A.; Indyk, P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In Proceedings of the Annual IEEE Symposium on Foundations of Computer Science (FOCS'06), Berkeley, CA, USA, 22–24 October 2006; Volume 47, pp. 459–468.
5. Zhang, D.; Wang, J.; Cai, D.; Lu, J. Self-taught hashing for fast similarity search. In Proceedings of the 33rd international ACM SIGIR Conference on Research and Development in Information Retrieval, Geneva, Switzerland, 19–23 July 2010; Association for Computing Machinery: New York, NY, USA, 2010; pp. 18–25.
6. He, K.; Wen, F.; Sun, J. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Portland, OG, USA, 25–27 June 2013; pp. 2938–2945.
7. Shen, F.; Zhou, X.; Yang, Y.; Song, J.; Shen, H.T.; Tao, D. A fast optimization method for general binary code learning. *IEEE Trans. Image Process.* **2016**, *25*, 5610–5621. [CrossRef] [PubMed]
8. Xu, H.; Wang, J.; Li, Z.; Zeng, G.; Li, S.; Yu, N. Complementary hashing for approximate nearest neighbor search. In Proceedings of the 2011 International Conference on Computer Vision (ICCV), Barcelona, Spain, 6–13 November 2011; pp. 1631–1638.
9. Wang, J.; Liu, W.; Kumar, S.; Chang, S.F. Learning to hash for indexing big data—A survey. *Proc. IEEE* **2015**, *104*, 34–57. [CrossRef]
10. Wang, J.; Zhang, T.; Sebe, N.; Shen, H.T. A survey on learning to hash. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 769–790. [CrossRef]

11. Roweis, S.T.; Saul, L.K. Nonlinear dimensionality reduction by locally linear embedding. *Science* **2000**, *290*, 2323–2326. [CrossRef]
12. Bai, X.; Yang, H.; Zhou, J.; Ren, P.; Cheng, J. Data-dependent hashing based on p-stable distribution. *IEEE Trans. Image Process.* **2014**, *23*, 5033–5046. [CrossRef]
13. Gong, Y.; Lazebnik, S.; Gordo, A.; Perronnin, F. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *35*, 2916–2929. [CrossRef]
14. Shen, F.; Shen, C.; Liu, W.; Shen, H.T. Supervised Discrete Hashing. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 37–45.
15. Cao, Z.; Long, M.; Wang, J.; Yu, P.S. Hashnet: Deep learning to hash by continuation. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 5608–5617.
16. Yang, H.F.; Lin, K.; Chen, C.S. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *40*, 437–451. [CrossRef]
17. Liu, H.; Wang, R.; Shan, S.; Chen, X. Deep supervised hashing for fast image retrieval. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 2064–2072.
18. Li, W.J.; Wang, S.; Kang, W.C. Feature learning based deep supervised hashing with pairwise labels. In Proceedings of the 2015 International Joint Conference on Artificial Intelligence (IJCAI), Buenos Aires, Argentina, 28 July–1 August 2015; pp. 1711–1717.
19. Li, Q.; Sun, Z.; He, R.; Tan, T. Deep supervised discrete hashing. In *Advances in Neural Information Processing Systems*; Curran Associates: New York, USA, 2017; pp. 2482–2491.
20. Wang, D.; Wang, L. On OCT image classification via deep learning. *IEEE Photonics J.* **2019**, *11*, 1–14. [CrossRef]
21. Wu, Y.; Sun, Q.; Hou, Y.; Zhang, J.; Zhang, Q.; Wei, X. Deep covariance estimation hashing. *IEEE Access* **2019**, *7*, 113223–113234. [CrossRef]
22. Cakir, F.; He, K.; Bargal, S.A.; Sclaroff, S. Hashing with mutual information. *IEEE Trans. Pattern Anal. Mach. Intell.* **2019**, *41*, 2424–2437. [CrossRef]
23. Li, N.; Li, C.; Deng, C.; Liu, X.; Gao, G. Deep joint semantic-embedding hashing. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, 13–19 July 2018; pp. 2397–2403.
24. Cheng, S.; Wang, L.; Du, A. An adaptive and asymmetric residual hash for fast image retrieval. *IEEE Access* **2019**, *7*, 78942–78953. [CrossRef]
25. Du, A.; Wang, L.; Cheng, S.; Ao, N. A Privacy-Protected Image Retrieval Scheme for Fast and Secure Image Search. *Symmetry* **2020**, *12*, 282. [CrossRef]
26. Cheng, S.L.; Wang, L.J.; Huang, G.; Du, A.Y. A privacy-preserving image retrieval scheme based secure kNN, DNA coding and deep hashing. *Multimed. Tools Appl.* **2019**. [CrossRef]
27. Li, J. Context-based image re-ranking for content-based image retrieval. In Proceedings of the 2011 IEEE Symposium on Computational Intelligence for Multimedia, Signal and Vision Processing, Paris, France, 11–15 April 2011; pp. 39–46.
28. Salakhutdinov, R.; Hinton, G. Semantic hashing. *Int. J. Approx. Reason.* **2009**, *50*, 969–978. [CrossRef]
29. Erin Liong, V.; Lu, J.; Wang, G.; Moulin, P.; Zhou, J. Deep hashing for compact binary codes learning. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 2475–2483.
30. Lin, K.; Lu, J.; Chen, C.S.; Zhou, J. Learning compact binary descriptors with unsupervised deep neural networks. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 26 June–1 July 2016; pp. 1183–1192.
31. Jin, S.; Yao, H.; Sun, X.; Zhou, S. Unsupervised semantic deep hashing. *Neurocomputing* **2019**, *351*, 19–25. [CrossRef]
32. Yang, E.; Deng, C.; Liu, T.; Liu, W.; Tao, D. Semantic structure-based unsupervised deep hashing. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18), Stockholm, Sweden, 13–19 July 2018; pp. 1064–1070.
33. Li, C.; Wang, L.; Cheng, S.; Ao, N. Generative Adversarial Network-Based Super-Resolution Considering Quantitative and Perceptual Quality. *Symmetry* **2020**, *12*, 449. [CrossRef]

34. Ghasedi Dizaji, K.; Zheng, F.; Sadoughi, N.; Yang, Y.; Deng, C.; Huang, H. Unsupervised deep generative adversarial hashing network. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Coimbatore, India, 21–22 September 2018; pp. 3664–3673.
35. Deng, C.; Yang, E.; Liu, T.; Li, J.; Liu, W.; Tao, D. Unsupervised semantic-preserving adversarial hashing for image search. *IEEE Trans. Image Process.* **2019**, *28*, 4032–4044. [CrossRef]
36. Huang, L.K.; Chen, J.; Pan, S.J. Accelerate learning of deep hashing with gradient attention. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 5271–5280.
37. Weiss, Y.; Torralba, A.; Fergus, R. Spectral hashing. In *Advances in Neural Information Processing Systems*; The MIT Press: Cambridge, MA, USA, 2009; pp. 1753–1760.
38. Heo, J.P.; Lee, Y.; He, J.; Chang, S.F.; Yoon, S.E. Spherical hashing. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Providence, RI, USA, 16–21 June 2012; pp. 2957–2964.
39. Jin, Z.; Li, C.; Lin, Y.; Cai, D. Density sensitive hashing. *IEEE Trans. Cybern.* **2013**, *44*, 1362–1371. [CrossRef] [PubMed]