

# Deep Position-Aware Hashing for Semantic Continuous Image Retrieval

Ruikui Wang<sup>1,2</sup>, Ruiping Wang<sup>1,2</sup>, Shishi Qiao<sup>1,2</sup>, Shiguang Shan<sup>1,2</sup>, Xilin Chen<sup>1,2</sup>

<sup>1</sup> Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS),  
Institute of Computing Technology, CAS, Beijing, 100190, China

<sup>2</sup> University of Chinese Academy of Sciences, Beijing, 100049, China

Frui kui . wang, shi shi . qi aoG@vi pl . i ct . ac . cn, Fwangruip i ng, sgshan, xl chenG@i ct . ac . cn

## Abstract

*Preserving the semantic similarity is one of the most important goals of hashing. Most existing deep hashing methods employ pairs or triplets of samples in training stage, which only consider the semantic similarity within a mini-batch and depict the local positional relationship in Hamming space, leading to intermittent semantic similarity preservation. In this paper, we propose Deep Position-Aware Hashing (DPAH) to ensure continuous semantic similarity in Hamming space by modeling global positional relationship. Specifically, we introduce a set of learnable class centers as the global proxies to represent the global information and generate discriminative binary codes by constraining the distance between data points and class centers. In addition, in order to reduce the information loss caused by relaxing the binary codes to real-values in optimization, we propose kurtosis loss (KT loss) to handle the distribution of real-valued features before thresholding to be double-peak, and then enable the real-valued features to be more binary-like. Comprehensive experiments on three datasets show that our DPAH outperforms state-of-the-art methods.*

## 1. Introduction

Owing to fast retrieval speed and low memory cost, learning to hash [11, 41, 31, 19, 16, 28, 25, 12, 35, 1] has been widely used in large-scale image retrieval. With CNN’s powerful representation capabilities, deep hashing methods [48, 49, 45, 36, 2, 13, 7, 3, 30, 47] have attracted more and more attention in recent years.

Generally speaking, hashing is a kind of function that maps data from high-dimensional real-valued space to low-dimensional Hamming space and keeps the semantic similarities as good as possible. Although the existing deep hashing methods, such as [43, 18, 22, 24, 21, 4, 37] have reached excellent performance on benchmarks, most of them can only give a local description to the positional relationship of samples in Hamming space. As shown in Figure

Figure 1. An illustration of Hamming positional relationship using different supervised manners. Different colors denote different categories.

1(a)(b), the supervised manner they adopt either belongs to pair-wise [41, 27, 32, 24, 51, 4] or triplet-wise [18, 50, 40] which only consider the local similarity within a mini-batch and has no communication between different batches. This will lead to intermittent semantic similarity from the global perspective, especially on multi-label retrieval where similarity among samples is more continuous and the task is more common in real world scenarios than single-label retrieval. A couple of works have tried to consider the global positional relationship in Hamming space, e.g. the point-wise methods [34, 22, 20, 44, 37]. They aim to classify in Hamming space using softmax loss, max-margin loss or L2-loss as the supervised signal. Nevertheless, their binary codes will not be further optimized as soon as the classification results are all correct, which leads to insufficient discriminability. This can be seen from Figure 1(c).

In order to preserve continuous semantic similarity as well as to enhance the discriminability of the binary codes, we introduce a kind of learnable class center as the global proxy of one category. This class center is like the origin of a coordinate system, and once the origin is fixed, then the coordinates of each data point are determined. In other words, the global position of all data points within one category will be aware. It is so-called position awareness (PA) whose learning process is vividly shown in Figure 2. Then we constrain the Hamming distance between data points to make the binary codes more discriminative. Specifically, a

sample should be close to its relevant class centers (e.g. a data point may belong to multiple categories in some scenarios) and be closer to them by a margin than to the nearest irrelevant class center. By doing so, the continuous semantic similarity can be characterized by different Hamming distance between data points and corresponding class centers. Figure 1(d) demonstrates the target state of our approach.

Apart from semantic similarity preservation, the optimization of discrete binary codes is another challenge in hash learning. As validated in most existing methods, to make the network trainable in an end-to-end manner, we choose to relax the discrete codes to real-values by replacing the threshold function sign by sigmoid. However, this relaxation incurs that the variance of the real-valued features before thresholding will become quite large, whose distribution can be seen from the blue area of Figure 3. Each half of the blue area will then be thresholded as discrete values (e.g. 0/1) later, which inevitably leads to heavy information loss. On the contrary, if one can constrain the distribution of the real-valued features to be within the red area in Figure 3, the information loss can be reduced intuitively. Based on such finding, we propose a novel kurtosis loss (K-T loss), a simple approach used for making real-valued features distribution be a steeper double-peak. In this way, the data structure before and after thresholding becomes more similar, and thus the information loss can be mitigated.

In summary, this paper has three main contributions:

- We explicitly model the global positional relationship in Hamming space to preserve the continuous semantic similarity and enhance the discriminability of binary codes by introducing a Hamming margin;
- Kurtosis loss is proposed to reduce the information loss caused by thresholding and further narrow the gap between real-valued space and desired Hamming space;
- Comprehensive experiments are conducted to show that our methodology significantly outperforms existing hashing methods for fast image retrieval in both single-label and multi-label databases.

## 2. Related Works

Hashing is a potent weapon for fast data retrieval. Compared with real-valued feature based methods [6, 9], hashing with binary codes is matching efficient and storage free. The classical literature contains the family of methods known as Locality Sensitive Hashing (LSH) [11] and its variants [31]. Comprehensive survey of traditional hashing methods can be found in [39].

More recently, in light of the progress of deep neural network, deep hashing methods gradually enter the front stage. Simultaneously learning image feature and hash function in an end-to-end manner is the key advantage of deep hashing

Figure 2. Position awareness learning: the data points' positions in Hamming space are rambling before learning, with the help of PA, they gradually become organized like a coordinate whose origins correspond the learned class centers. In the figure, different color dots denote data points from different class. The dashed pentagrams and solid pentagrams denote learnable class centers and learned class centers respectively. The black dashed lines denote the data points' position vectors.

over traditional methods. Most deep hashing methods are analogous in terms of feature learning module, hash function learning manner reflects difference instead. Deep based supervised hashing methods can be divided into four categories from the perspective of semantic similarity preservation.

**Pair-wise.** This kind of methods intend to pull the codes of similar images together and push the codes of dissimilar images away from each other. This idea is widely used in metric learning. Representative methods include, ADSH [14], DPSH [21], COSDISH [15], DHN [51], DSH [24] and HashNet [4]. The commonality of these methods are that they do not require fully supervised information and only needs an affinity matrix. However, they need to generate dense pairs online, which results in much consumption of training time. On the other hand, since deep learning is a batch-based training method, there may be no communication between different batches, which results in insufficient sample mining. In addition, from the perspective of binary feature distribution, this type of methods only consider the relative positional relationship between sample pairs, so that the original semantic relevance is not well preserved.

**Triplet-wise.** Compared to the previous one, this type of methods consider the local positional relationship within a sample triplet. Representative methods include DNNH [18], DSRBH [50], DTSH [40]. Similar with the pair-wise method, this one require online generation of triplets, and even hard negative mining during training, which costs a lot of time. And only the local position relationship between samples is considered.

**List-wise.** This kind of methods are based on learning to rank. Representative methods include [5], [38], [46]. They decompose a sort of retrieved samples into multiple triplets for processing, which is more elaborate than the triplet-wise, but still inherits the lack of triplet-wise.

**Point-wise.** Different from the several previous methods, this type of methods need no sample pairs or triplets, only

need aware label information for each sample to guide the learning of the hash functions. This trait make it more scalable to large-scale data retrieval [44]. Representative works include DLBHC [22], SSDH [44] and Greedy Hash [37]. Compared with the above approaches, this kind of methods can describe the global information with the classification hyperplanes optimized adaptively as the training data flows and our DPAH belongs to point-wise.

Minimizing quantization error is another important challenge in learning to hash. ITQ [12] minimizes quantization error by finding an optimal rotation matrix; SDH [34], ADSH [14] and other methods optimize the binary code discretely, so that there is no quantization error; DPSH [21] and DSH [24] control the quantization error by imposing a regularizer; HashNet [4] approximates the sign function by incrementing the slope of the tanh function; Greedy hash [37] decreases the gap between binary coding learning and feature representation learning by redefining the back propagation of the sign function. This paper novelly reduces the gap by constraining the feature distribution of the real-valued space before Hamming space. Since there is no direct manipulation of discrete binary features, complex and time-consuming column-by-column discrete optimization processes are avoided, which is very beneficial for end-to-end optimization.

### 3. Approach

Our goal is to learn discriminative binary codes for images such that: (a) the global continuous semantic similarity should be preserved in Hamming space, i.e. the more semantic information shared by two images, the closer they will be in Hamming space; (b) the information loss before and after thresholding should be small. The training pipeline is described as follow: taking into the training images from different classes, our method first extracts the real-valued features with the stacked convolution layers and several fully connected layers. Then with the help of the PA (Position Awareness) module, the global positional relationships among images are built to ensure continuous semantic similarity in Hamming space. At the same time, the information loss is reduced by adjusting the distribution of real-valued features with the proposed KT loss. Finally, the thresholded outputs are quantized to generate binary codes for these images.

#### 3.1. Hamming Position Awareness

Given  $N$  tuples of training points  $\{I_i, y_i\}_{i=1}^N$ , each represented by image  $I_i \in \mathbb{R}^{H \times W \times 3}$  and a  $C$ -dimensional label vector  $y_i \in \{0, 1\}^C$  (either one-hot or multi-labeled), we aim to learn a mapping from the RGB space to Hamming space:  $f: \mathbb{R}^{H \times W \times 3} \rightarrow \{0, 1\}^K$ , which encodes  $I_i$  into  $K$ -bit binary code  $b_i$  such that the more semantic information shared by two images, the closer they will be

in Hamming space. Firstly, we model the global relationship by introducing  $C$  class centers  $\{c_i\}_{i=1}^C (c_i \in \{0, 1\}^K)$  which are learnable during training, i.e. they can record the historical data information and be updated dynamically driven by the new data, thus it is suitable to capture the global information of categories. Secondly, we embed continuous semantic similarity by constraining the distance between data points and class centers. Specifically, if  $I_i$  belongs to one category  $c_i$ , then corresponding  $b_i$  should be very close to  $c_i$  and closer to it by a margin than the nearest irrelevant class center. Furthermore, if  $I_i$  is multi-labeled, i.e.  $\text{label}\{i\} = \{c_{i_1}, c_{i_2}, \dots, c_{i_{n_i}}\}$ , then corresponding  $b_i$  should be very close to the center  $c_{i_j}$  of the label set i.e.  $c_{i_j} = \frac{1}{n_i} \sum_{j \in \text{label}\{i\}} c_j$  and closer to it by a margin than the nearest one which is excluded from this label set. It is not difficult to note that the single-label case is a special example of the multi-label case mentioned above. Such constraints can be formulated as:

$$L_{\text{inter}}^i = \max\{d_H(b_i, c_{i_j})(1 + \gamma) - \min_{j' \in \text{label}\{i\}} d_H(b_i, c_{i_{j'}}), 0\} \quad (1)$$

s.t.  $b_i, c_i \in \{0, 1\}^K, i \in \{1, \dots, N\}$

where  $L_{\text{inter}}^i$  denotes inter-class loss,  $d_H(\cdot, \cdot)$  is the Euclidean distance, defined as  $d_H(x, y) = \frac{1}{2} \|x - y\|_2^2$ , which is equivalent to Hamming distance between two binary codes,  $\gamma$  is the margin hyperparameter, and  $n_i$  denotes the size of the label  $\{i\}$ . It is difficult to optimize directly with back propagation algorithm since the  $\min(\cdot)$  and  $\max(\cdot)$  functions will easily cause the loss instable. Hence we turn to optimize the smooth upper bound of Eq.(1) which is magnified as:

$$L_{\text{inter}}^i = -\log\left\{\frac{e^{(-d_H(b_i, c_{i_j})(1 + \gamma))}}{e^{(-d_H(b_i, c_{i_j})(1 + \gamma))} + \sum_{j' \in \text{label}\{i\}} e^{(-d_H(b_i, c_{i_{j'}}))}}\right\} \quad (2)$$

s.t.  $b_i, c_i \in \{0, 1\}^K$

The detailed derivation from Eq.(1) to Eq.(2) can be found in Appendix A. At the same time, considering the discriminability in Hamming space and the position compactness within one class clusters, we impose a tight constraint like [42] as Eq.(3).

$$L_{\text{intra}}^i = \frac{1}{2} \|b_i - c_{i_j}\|_2^2 \quad (3)$$

s.t.  $b_i, c_i \in \{0, 1\}^K$

In Eq.(3),  $L_{\text{intra}}^i$  denotes the intra-class loss. Actually the idea of intra compactness has been widely used in many fields, such as metric learning, face recognition. Our method also contains the idea of intra compactness. Different from [42], we consider both the intra compactness and inter discriminability, i.e. intra-class samples are clos-

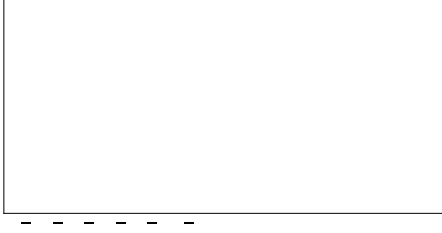


Figure 3. Distribution of real-valued outputs before thresholding.

er and inter-class samples will be kept away from a certain distance by introducing the margin  $\gamma$ .

The above two loss functions will push each binary code to its reasonable position and gradually converge to the state shown in Figure 1(d). Combining the Eq.(2) with Eq.(3), we get Eq.(4) as follows:

$$L_{dis} = \frac{1}{N} \sum_{i=1}^N (L_{inter}^i + L_{intra}^i) \quad (4)$$

$$\text{s.t. } b_i \in \{0, 1\}^K$$

However, directly optimizing Eq.(4) in an end-to-end manner is infeasible since the binary constraints will make it intractable to train the network with back propagation algorithm. To this end, a natural scheme is used by relaxing the integer constraint of  $b_i$  to real constraint. Specifically, we adopt sigmoid as the threshold function to approximate sign, therefore  $b_i$  can be relaxed as  $b_i = \sigma(u_i)$ . Here  $\sigma(\cdot)$  and  $u_i$  denote the sigmoid and real-valued output of the last fully connected layer respectively. Then Eq.(4) is rewritten as :

$$L_{dis} = \frac{1}{N} \sum_{i=1}^N (L_{inter}^i + L_{intra}^i) \quad (5)$$

$$\text{s.t. } b_i \in (0, 1)^K$$

In order to reduce the cost of relaxation, as advocated in [44], an additional regularizer is imposed on  $b_i$  as:

$$L_{reg} = \frac{1}{2N} \sum_{i=1}^N \left( \text{mean}(b_i) - 0.5 \right)^2 - \frac{1}{2N} \sum_{i=1}^N \left( b_i - 0.5 \right)^2 \quad (6)$$

$$\text{s.t. } b_i \in (0, 1)^K$$

where the first term is used for making the hash bits balanced, and the second one is for controlling the quantization error.

### 3.2. Kurtosis Loss

The similarity would be well preserved with Eq.(5). However, some issues still exist caused by the approximation of the threshold function adopted in Eq.(5) which

has not been noticed in previous works. Specifically, we find that as the training progresses, the variance of the real-valued network output (i.e.  $u_i$ ) will gradually increase and its distribution is present in the blue area in Figure 3. Each half of the blue area will be thresholded as discrete values (e.g. 0/1) later, which inevitably leads to information loss, i.e. the large variance of the half of the blue area cannot be preserved after thresholding. In order to minimize such information loss, directly optimizing the discrete binary codes is generally NP-hard [34], we turn to constrain the variance of  $u_i$  to be smaller. In specific, a threshold  $t$  is set, once the maximum value of  $u_i$  exceeds the threshold, it will cause loss. We form it as follows:

$$L_{kti} = \frac{1}{2N} \sum_{i=1}^N \sum_{k=1}^K [\max^2(u_i(k) - t, 0) + \max^2(-u_i(k) - t, 0)] \quad (7)$$

It is not difficult to realize from the curve of the Eq.(7) that the real-valued feature will be pulled back with  $L_{kti}$ , which presents a form, larger kurtosis of  $u_i$ . The new distribution of  $u_i$  is shown as the red area which is steeper than the blue area in Figure 3. In mathematics, kurtosis is used to describe the steepness of a distribution. Therefore we call  $L_{kti}$  kurtosis loss. By doing so, the distribution structure of features before thresholding is much like the one after thresholding, and thus the information loss and the gap before and after thresholding can be reduced. It should be noted that we impose an additional regularize on features before thresholding, which is different from [24, 21] whose goal is to constrain the real-valued feature to be binary-like.

Now, we rewrite the overall loss function as follows:

$$L = L_{dis} + \lambda L_{reg} + \mu L_{kti} \quad (8)$$

$$\text{s.t. } b_i \in (0, 1)^K$$

where  $\lambda$  and  $\mu$  are pre-defined weighting parameters to balance the  $L_{reg}$  and  $L_{kti}$ .

### 3.3. Discussions

**Comparison with metric learning** As mentioned in Introduction section, semantic similarity preservation is one of the challenges in hashing. Seeking intra compactness and inter separability, metric learning is a nice choice for solving such problem. It has been widely adopted in many existing hashing methods, such as pair-wise methods, triplet-wise methods which have been thoroughly introduced in related work section. Different from point-wise metric learning methods [42, 26, 10], our DPAH unifies single-label retrieval and multi-label retrieval in one formulation and shows distinctive superiority in multi-label scenarios, which is verified in later experiment section. Furthermore, optimization in discrete Hamming space is an extra challenge in hashing compared to metric learning of which the analysis lies in Euclidean space. To this end, we propose KT loss

to constrain the feature distribution to be double-peak and devote to alleviate such trouble.

**Comparison with existing point-wise hashing methods.** As mentioned in related work section, our DPAH belongs to point-wise type, which is more practical in real applications. Most recent point-wise hashing methods, such as SDH [34], SSDH [44], Greedy Hash [37] use inner product as metrics in Hamming space. Our DPAH adopts Euclidean distance which is equivalent to Hamming distance instead. There are two advantages using this scheme: firstly, Euclidean distance in binary space has an intuitive meaning and can directly reflect the role of optimization; secondly, it becomes easier to introduce margin to enhance the discriminability of binary code. Extensive experiments demonstrate our DPAH has clear superiority over those point-wise hashing methods.

## 4. Experiments

### 4.1. Datasets and Experimental Settings

We conduct experiments on three datasets: (1) **ImageNet** [33] consists of about 1.2M images belonging to 1,000 mutually exclusive categories. Following the same setting in [4], we randomly select 100 categories and use all images of these categories in training set as the database, using all images of these categories in validation set to form query set. Moreover, 130 images per class randomly selected from the database are used to train the deep hashing network. (2) **MS COCO** [23] contains 82,783 training images and 40,504 validation images. Following the same setting in [4], we randomly select 5,000 images to form the query set, and the rest as the database. Further, 10,000 images were randomly selected from the database as the training set. (3) **NUS-WIDE** [8] contains 269,648 images collected from Flickr and associated with 81 concepts. Following the similar protocols as [48], we randomly select 2,100 images from 21 most happened semantic labels as the query set and the rest as the training set.

Same as most of the previous hashing methods, the ground truth is defined by class-level labels. For ImageNet, images from the same class are considered semantically relevant and vice versa. For MS COCO and NUS-WIDE, if two images share at least one positive label, they are considered relevant, and irrelevant otherwise. As for evaluation metrics, we use the mean Average Precision (mAP) for different code lengths and precision-recall curves (48-bit). For fair comparison with state-of-the-art, we use mAP@1K for ImageNet, mAP@5K for MS COCO and mAP@50K for NUS-WIDE respectively. Our DPAH method is implemented with Pytorch [29] framework<sup>1</sup>. The AlexNet [17] is adopted as our backbone. During training, we set the batch size as 256, momentum as 0.9, weight decay as 5e-4. The

<sup>1</sup>Our source codes are available at <http://vipl.ict.ac.cn/resources/codes>.

Method	NUS-WIDE		MS COCO		ImageNet	
	mAP@50K	mAP	mAP@5K	mAP	mAP@1K	mAP
contrastive loss	0.7056	0.6414	0.6618	0.5963	N/C	N/C
triplet loss	0.7916	0.7451	0.7183	0.6406	0.5710	0.4655
maximum-margin	0.7393	0.6735	0.7193	0.5654	N/C	N/C
softmax+CE	0.7544	0.6736	0.7334	0.5770	0.7014	0.6218
softmax+PA	0.8038	0.7171	0.7549	0.6266	0.7014	0.6218
HPA( = 0)	0.8279	0.7591	0.7731	0.6634	0.7027	0.6316
HPA( = 0.01)	<b>0.8307</b>	<b>0.7634</b>	<b>0.7756</b>	<b>0.6682</b>	<b>0.7050</b>	<b>0.6334</b>

Table 1. Comparison of retrieval performance with baselines. The results are obtained with 48-bit binary codes.

(a) NUS-WIDE (b) MS COCO (c) ImageNet

Figure 4. Comparison of the model with and without KT loss on three databases under different code lengths.

learning rate is fixed as 0.001. We train DPAH model 200 epochs in total.

### 4.2. Evaluation of Hamming Position Awareness

In order to verify the effectiveness of PA module, we compare it with several methods corresponding to different supervised manner: (1) contrastive loss, a representative implementation of pair-wise method; (2) triplet loss, a representative implementation of triplet-wise method; (3) softmax+CE [44], softmax followed by cross entropy loss which is a representative implementation of point-wise method; (4) maximum-margin, which is proposed in [44] and reproduced carefully by us; (5) softmax+PA, softmax followed by PA module in which the metric is inner product, consisting with softmax; (6) our Hamming Position Awareness (HPA) module with different hyperparameter, i.e. . Without loss of generality, we only test the case with 48-bit and set = 1 and = 0.2 in our model according to Section 3.1. Note that we aim to verify the effectiveness of PA independently, therefore, no KT loss is used in this section. The performance of all methods are listed in Table 1. The N/C in Table 1 denotes the model cannot converge under the same experimental settings with others.

We have four observations from the comparison table: **First**, with the help of PA module, HPA and softmax+PA are generally better than their corresponding baseline methods on the three databases, which indicates that our PA module is very effective for the semantic similarity preservation in Hamming space; **Second**, the superiority of HPA is obvious compared with baselines especially on multi-label databases, which verified our argument proposed in introduction. Generally speaking, multi-label data means more complex semantic similarities. Based on the distance constraint, our PA can measure the degree of similarity, i.e. the

Method	VGG16		ResNet50		ResNet101	
	mAP@1K	mAP	mAP@1K	mAP	mAP@1K	mAP
softmax	0.8327	0.7729	0.8134	0.7375	0.8370	0.7717
softmax+KT loss	0.8438	0.7929	0.8302	0.7613	0.8528	0.7938
DPAH\KT loss	0.8449	0.7980	0.8516	0.7944	0.8710	0.8266
DPAH	<b>0.8487</b>	<b>0.8063</b>	<b>0.8566</b>	<b>0.8029</b>	<b>0.8765</b>	<b>0.8361</b>

Table 2. Retrieval performance of the model with KT loss under different backbones. The results are obtained with 48-bit binary codes on ImageNet.

(a) without KT loss

(b)  $t = 15$

(c)  $t = 10$

(d)  $t = 5$

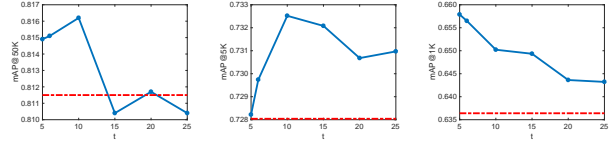
Figure 5. Distribution of real-valued network outputs under different settings of  $t$ .

number for categories shared by two images. This can be further demonstrated in visualization part of Section 4.4. On the contrary, pair-wise and triplet-wise methods (i.e. the first two rows of Table 1) are based solely on a similarity matrix so that they cannot distinguish pairs whether they share one or more categories. **Third**, from the fourth row and second last row of the Table 1, we can see that softmax+CE is not performing as well as HPA. This may result from that softmax+CE focuses on classification accuracy leading to insufficient discriminability. **Fourth**, from the last two rows of the Table 1, we notice that the retrieval performance will further increase with Eq.(3), demonstrating the effectiveness of making intra-class compact.

### 4.3. Evaluation of the KT loss

In this part, we evaluate KT loss from the following three aspects: multiple datasets, different network structures and parameter sensitivity. Firstly, we test the role of KT loss on three datasets (i.e. both single-label and multi-label dataset). Then, we test the performance of KT loss with VGG16, ResNet50 and ResNet101 as backbone respectively. Finally, 16-bit is taken as an example to test the sensitivity of the hyperparameter  $t$ . As mentioned in Section 3.2, the variance of the network real-valued output will gradually increase without KT loss, we use the maximum value it can reach as the upper bound of the hyperparameter  $t$ .

Firstly, we can see from Figure 4 that KT loss will generally improve the model’s retrieval ability compared with the baselines on all datasets; Secondly, from Table 2, in which

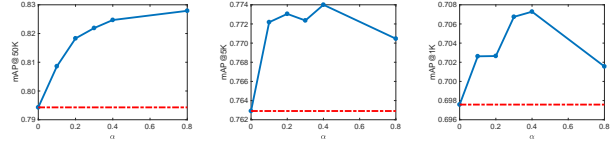


(a) NUS-WIDE

(b) MS COCO

(c) ImageNet

Figure 6. The sensitivity to  $t$ . The dotted line indicates the result without KT loss.



(a) NUS-WIDE

(b) MS COCO

(c) ImageNet

Figure 7. The sensitivity to  $\alpha$ . The dotted line indicates the result without margin.

DPAH\KT loss stands for dropping KT loss from DPAH, we observe that KT loss have a stable effect with different backbones. The above two experiments show that KT loss is robust. Recalling that KT loss does not utilize supervisory information, and it can theoretically be used for different discriminative losses, such as softmax. The first two rows of Table 2 shows that KT loss also have obvious effects on softmax loss, thus indicating the versatility of KT loss; Thirdly, Figure 6 reveals that KT loss is not very sensitive to hyperparameter  $t$ . The retrieval performances can be improved stably when setting  $t$  under a reasonable range. (e.g. [5, 10]). These quantitative results validate that KT loss reduces information loss caused by the approximation of the threshold function. As shown in Figure 5, the smaller  $t$  is, the larger the kurtosis of the real-valued network output is, and the smaller the information loss is. However, when  $t$  is too small, the discriminability of binary codes will be affected. In other words, we should make a tradeoff between information loss and discriminability. Fortunately, the reasonable range of hyperparameter  $t$  is relatively large.

### 4.4. Empirical Analysis

**Parameter Sensitivity Analysis:** In the field of metric learning, the use of the margin in the training of the model tends to increase its generalization ability. Inspired by this, we believe that once the position of binary codes in Hamming space is aware, imposing a margin allows it to further consolidate its position (the position becomes more aware), thereby increasing the generalization ability of the retrieval model. To verify this, we test the impact of different values of the margin on three databases. It can be found from the Figure 7, compared to the baseline, the performance is significantly improved after the margin is applied. But when the margin is too large, optimal hash functions are difficult to learn.

**Visualization of Hash Codes and Retrieval Samples:**

Method	NUS-WIDE				MS COCO				ImageNet			
	16-bit	32-bit	48-bit	64-bit	16-bit	32-bit	48-bit	64-bit	16-bit	32-bit	48-bit	64-bit
LSH [11]	0.3915	0.4169	0.4114	0.4323	0.4118	0.4689	0.4992	0.5076	0.0699	0.1562	0.2205	0.2732
ITQ [12]	0.5754	0.5840	0.5873	0.5943	0.6293	0.6654	0.6793	0.6903	0.3245	0.4655	0.5219	0.5539
SDH [34]	0.7019	0.7158	0.7157	0.7256	0.5447	0.5857	0.6025	0.6127	0.4001	0.5515	0.6196	0.6516
KSH [25]	0.5693	0.5736	0.5754	0.5822	0.5924	0.6180	0.6345	0.6422	0.3600	0.4803	0.5327	0.5544
SITQ [12]	0.6312	0.6693	0.6808	0.6888	0.6300	0.6760	0.7047	0.7163	0.3250	0.4750	0.5369	0.5779
DSH [24]	0.7014	0.7275	0.7261	0.7289	0.6218	0.6292	0.6383	0.6380	0.4526	0.5563	0.6062	0.6235
HashNet [4]	0.7260	0.7704	0.7797	0.7731	0.6578	0.7121	0.7316	0.7374	0.4643	0.5925	0.6558	0.6544
DHN [51]	0.7443	0.7490	0.7486	0.7482	0.6888	0.7158	0.7221	0.7274	0.2671	0.4367	0.4933	0.5347
DNNH [18]	0.7847	0.8002	0.8053	0.8062	0.6732	0.7137	0.7298	0.7362	0.4946	0.5805	0.6035	0.6143
SSDH [44]	0.7188	0.7225	0.7393	0.7460	0.6970	0.7250	0.7410	0.7440	0.6342	0.6915	0.7014	0.7069
DPAH	<b>0.8162</b>	<b>0.8266</b>	<b>0.8346</b>	<b>0.8280</b>	<b>0.7325</b>	<b>0.7675</b>	<b>0.7777</b>	<b>0.7815</b>	<b>0.6517</b>	<b>0.7001</b>	<b>0.7149</b>	<b>0.7138</b>

Table 3. Comparison of retrieval performance of our DPAH method and the other hashing methods on three benchmark datasets.

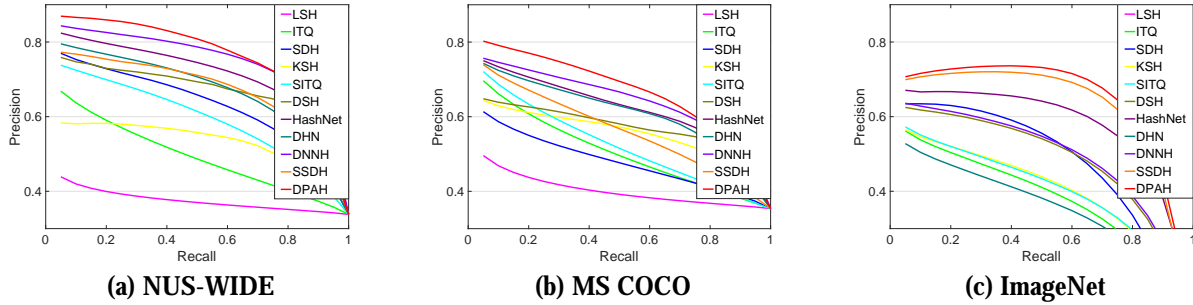


Figure 8. Precision-recall curves (48-bit) of our DPAH method and the other hashing methods on three benchmark datasets.

#### 4.5. Comparison with the State-of-the-arts

**Comparative methods:** We compare DPAH with ten classical hashing methods: unsupervised methods LSH [11], ITQ [12], supervised shallow methods SITQ [12], KSH [25], SDH [34] and supervised deep methods DNNH [18], DHN [51], DSH [24], HashNet [4], SSDH [44]. In order to compare more recent methods, we also made a supplementary experiment on CIFAR-10 in the Appendix B. For all shallow hashing methods, we use the fc7 layer of the AlexNet [17] pre-trained on ImageNet2012 as input. For all deep hashing methods, we use the AlexNet model pre-trained on ImageNet2012 as the backbone. In our DPAH, we set the hyperparameter  $t$  as 10, as 0.2, as 0.01, as 1 and as 0.01 respectively.

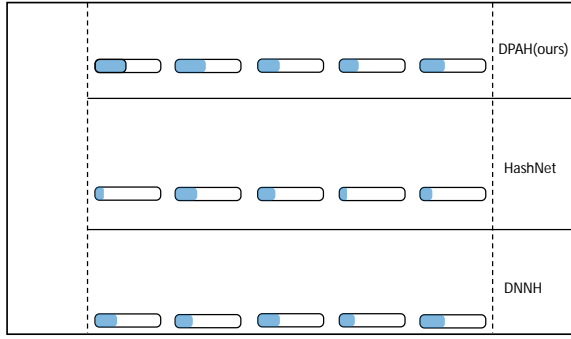
**Results:** Table 3 shows the retrieval performance comparison of our method against the others and Figure 8 gives the precision recall curves on three datasets with 48-bit binary codes. In general, supervised hashing method perform better than unsupervised methods, validating the importance of supervised information for learning discriminative binary codes. In addition, those CNN-based methods outperform the conventional hashing methods with deep features on both datasets by a large margin, suggesting that learning discriminative image representations and compact binary codes simultaneously in an end-to-end manner is ad-

(a) DNNH (b) HashNet (c) DPAH

Figure 9. Comparison of 48-bit Hamming space visualization of DNNH, HashNet and our DPAH.

With the help of t-SNE, we visualize the 48-bit binary codes generated by DNNH, HashNet and DPAH on ImageNet in Figure 9 (for ease of distinction, we sample 20 categories). These methods belong to triplet-wise, pair-wise and point-wise respectively. It can be seen from Figure 9(a) that the binary codes' position from different class is overlapped seriously, which results in poor discriminative ability. Then the overlap is weakened by HashNet in that the weight of data pairs is considered. Finally, we can see from Figure 9(c) that binary codes from different categories are well separated and vice versa, which validates that DPAH can effectively preserve semantic similarity in Hamming space. Apart from this, We can see from the Figure 10 and Figure 11 that our DPAH tends to return the images as relevant as possible.





(a) Case:1

(b) Case:2

Figure 10. Two retrieval cases on MS COCO, only the top-5 feedbacks are shown due to space limitation. Results were obtained with 64-bit binary codes. The floating point number with blue background below the image indicates the Jaccard similarity between sample and the query. The larger floating number indicates the more relevant the query is with retrieval sample.

(a) Case:1

(b) Case:2

Figure 11. Two retrieval cases on NUS-WIDE, only the top-5 feedbacks are shown due to space limitation. Results were obtained with 64-bit binary codes. The floating point number with blue background below the image indicates the Jaccard similarity between sample and the query. The larger floating number indicates the more relevant the query is with retrieval sample.

vantageous. Among the CNN-based methods, HashNet gets the best performance in pair-wise methods. It is because HashNet weights the training pairs that this method is more robust to deal with imbalanced similarity data; DNNH samples triplet online, although the training time is relatively long, the sequence relationship of the data can be guaranteed, which is more consistent with the target of the image retrieval. It achieves a near performance as HashNet; It is worth noting that SSDH, as a representative method of point-wise type, even outperforms HashNet on the single-label database, e.g. ImageNet, but not as good as HashNet on multi-label databases, e.g. MS COCO or NUS-WIDE. Our DPAH models global positional relationships to preserve continuous semantic similarities, showing superior performance on both single-label and multi-label databases.

## 5. Conclusion

In this paper, we propose a novel deep hashing framework named DPAH for image retrieval task. We attribute the promising performance to two aspects: **First**, the pro-

posed Position awareness module that ensures continuous semantic information in binary codes; **Second**, the novel KT loss for reducing the information loss between the real-valued feature space and the desired Hamming space. Since DPAH is a relatively general hashing method, it has wide potential applications in other tasks like information retrieval.

**Acknowledgements.** This work is partially supported by Natural Science Foundation of China under contracts Nos. 61922080, U19B2036, 61772500, and CAS Frontier Science Key Research Project No. QYZDJ-SSWJSC009.

## References

- [1] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt. Practical and optimal lsh for angular distance. In *Advances in Neural Information Processing Systems*, pages 1225–1233, 2015.
- [2] F. Cakir, K. He, S. Adel Bargal, and S. Sclaroff. Mihash: On-line hashing with mutual information. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 437–445, 2017.



- [3] F. Cakir, K. He, and S. Sclaroff. Hashing with binary matrix pursuit. In *Proceedings of the European Conference on Computer Vision*, pages 332–348. Springer, 2018.
- [4] Z. Cao, M. Long, J. Wang, and P. S. Yu. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5608–5617, 2017.
- [5] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136. ACM, 2007.
- [6] K. Chatfield, R. Arandjelović, O. Parkhi, and A. Zisserman. On-the-fly learning for visual search of large-scale image and video datasets. *International Journal of Multimedia Information Retrieval*, 4(2):75–93, 2015.
- [7] Z. Chen, X. Yuan, J. Lu, Q. Tian, and J. Zhou. Deep hashing via discrepancy minimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6838–6847, 2018.
- [8] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, page 48. ACM, 2009.
- [9] E. J. Crowley, O. M. Parkhi, and A. Zisserman. Face painting: querying art with photos. In *Proceedings of the British Machine Vision Conference*, pages 65.1–65.13, 2015.
- [10] J. Deng, J. Guo, N. Xue, and S. Zafeiriou. Arcface: Additive angular margin loss for deep face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4690–4699, 2019.
- [11] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.
- [12] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, 2013.
- [13] K. He, F. Cakir, S. Adel Bargal, and S. Sclaroff. Hashing as tie-aware learning to rank. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4023–4032, 2018.
- [14] Q.-Y. Jiang and W.-J. Li. Asymmetric deep supervised hashing. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [15] W.-C. Kang, W.-J. Li, and Z.-H. Zhou. Column sampling based discrete supervised hashing. In *Proceedings of the Thirtieth AAAI conference on Artificial Intelligence*, 2016.
- [16] W. Kong and W. jun Li. Isotropic hashing. In *Advances in Neural Information Processing Systems*, pages 1646–1654. 2012.
- [17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.
- [18] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3270–3278, 2015.
- [19] P. Li, A. Shrivastava, J. L. Moore, and A. C. König. Hashing algorithms for large-scale learning. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2011.
- [20] Q. Li, Z. Sun, R. He, and T. Tan. Deep supervised discrete hashing. In *Advances in Neural Information Processing Systems*, pages 2482–2491. 2017.
- [21] W. J. Li, S. Wang, and W. C. Kang. Feature learning based deep supervised hashing with pairwise labels. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.
- [22] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition workshops*, pages 27–35, 2015.
- [23] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. Microsoft coco: Common objects in context. 2014.
- [24] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2016.
- [25] W. Liu, J. Wang, R. Ji, Y. G. Jiang, and S. F. Chang. Supervised hashing with kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2012.
- [26] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song. Sphereface: Deep hypersphere embedding for face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 212–220, 2017.
- [27] M. Norouzi and D. M. Blei. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th International Conference on Machine Learning*, pages 353–360. ACM, 2011.
- [28] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov. Hamming distance metric learning. In *Advances in Neural Information Processing Systems*, pages 1061–1069, 2012.
- [29] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.
- [30] Q. Qiu, J. Lezama, A. Bronstein, and G. Sapiro. Foresthash: Semantic hashing with shallow random forests and tiny convolutional networks. In *Proceedings of the European Conference on Computer Vision*, pages 432–448. Springer, 2018.
- [31] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Advances in Neural Information Processing Systems*, pages 1509–1517, 2009.
- [32] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *Proceedings of the European Conference on Computer Vision*, pages 876–889. Springer, 2012.
- [33] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

- [34] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 37–45, 2015.
- [35] A. Shrivastava and P. Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems*, pages 2321–2329, 2014.
- [36] S. Su, G. Chen, X. Cheng, and R. Bi. Deep supervised hashing with nonlinear projections. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 2786–2792, 2017.
- [37] S. Su, C. Zhang, K. Han, and Y. Tian. Greedy hash: Towards fast optimization for accurate hash coding in cnn. In *Advances in Neural Information Processing Systems*, pages 806–815, 2018.
- [38] J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang. Learning hash codes with listwise supervision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3032–3039, 2013.
- [39] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.
- [40] X. Wang, Y. Shi, and K. M. Kitani. Deep supervised hashing with triplet labels. In *Asian Conference on Computer Vision*, pages 70–84. Springer, 2016.
- [41] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems*, 2008.
- [42] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *Proceedings of the European Conference on Computer Vision*, pages 499–515. Springer, 2016.
- [43] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.
- [44] H.-F. Yang, K. Lin, and C.-S. Chen. Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(2):437–451, 2018.
- [45] T. Yao, F. Long, T. Mei, and Y. Rui. Deep semantic-preserving and ranking-based hashing for image retrieval. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 3931–3937, 2016.
- [46] Z. Yu, F. Wu, Y. Zhang, S. Tang, J. Shao, and Y. Zhuang. Hashing with list-wise learning to rank. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 999–1002. ACM, 2014.
- [47] X. Yuan, L. Ren, J. Lu, and J. Zhou. Relaxation-free deep hashing via policy gradient. In *Proceedings of the European Conference on Computer Vision*, pages 134–150. Springer, 2018.
- [48] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Transactions on Image Processing*, 24(12):4766–4779, 2015.
- [49] Z. Zhang, Y. Chen, and V. Saligrama. Efficient training of very deep neural networks for supervised hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1487–1495, 2016.
- [50] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1556–1564, 2015.
- [51] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016.