Deep Self-Adaptive Hashing for Image Retrieval

Qinghong Lin[1], Xiaojun Chen[1✉], Qin Zhang[1], Shangxuan Tian[2], Yudong Chen[3]

[1]Shenzhen University, [2]Tencent, [3]The University of Queensland

linqinghong@email.szu.edu.cn,xjchen@szu.edu.cn,qinzhang2014@gmail.com

tianshangxuan@u.nus.edu,ydchen7@foxmail.com

ABSTRACT

Hashing technology has been widely used in image retrieval due to its computational and storage efficiency. Recently, deep unsupervised hashing methods have attracted increasing attention due to the high cost of human annotations in the real world and the superiority of deep learning technology. However, most deep unsupervised hashing methods usually pre-compute a similarity matrix to model the pairwise relationship in the pre-trained feature space. Then this similarity matrix would be used to guide hash learning, in which most of the data pairs are treated equivalently. The above process is confronted with the following defects: 1) The pre-computed similarity matrix is inalterable and disconnected from the hash learning process, which cannot explore the underlying semantic information. 2) The informative data pairs may be buried by the large number of less-informative data pairs. To solve the aforementioned problems, we propose a **Deep Self-Adaptive Hashing (DSAH)** model to adaptively capture the semantic information with two special designs: **Adaptive Neighbor Discovery (AND)** and **Pairwise Information Content (PIC)**. Firstly, we adopt the AND to initially construct a neighborhood-based similarity matrix, and then refine this initial similarity matrix with a novel update strategy to further investigate the semantic structure behind the learned representation. Secondly, we measure the priorities of data pairs with PIC and assign adaptive weights to them, which is relies on the assumption that more dissimilar data pairs contain more discriminative information for hash learning. Extensive experiments on several datasets demonstrate that the above two technologies facilitate the deep hashing model to achieve superior performance.

CCS CONCEPTS

• **Information systems** → **Novelty in information retrieval**.

KEYWORDS

Image Retrieval; Deep Unsupervised Hashing

1 INTRODUCTION

With the rapid development of the Internet, the explosive growth of multimedia data poses huge challenges to accurate nearest-neighbor based searching methods. Instead, due to its high efficiency, the approximate nearest neighbor (ANN) [15] based search methods have attracted increasing attention. Among them, hashing technologies contribute remarkably to their fast query speed and low storage overhead.

To be specific, hashing techniques can be divided into supervised and unsupervised categories. Supervised hashing [4, 11, 16, 17, 28, 34, 34, 41, 42] methods use the label information to train hashing models, which achieve fine performance. However, the annotations are highly time-consuming and also expensive to collect, which limits these methods in many practical applications. Due to this scenario, unsupervised hashing methods [10, 12, 13, 18, 32, 44] have drawn a large amount of attention. Many unsupervised hashing methods have been proposed in the past decade, including Spectral Hashing (SH) [46], Hashing with Graphs (AGH) [25], Iterative Quantization (ITQ) [10], Stochastic Generative Hashing (SGH) [6] , etc. The previous unsupervised hashing methods have made progress in this area, however, they normally adopt shallow architectures and severely depend on hand-crafted features (such as SIFT features [27]), which degrade the learning performance.

In recent years, numerous deep learning techniques have been introduced into unsupervised hashing methods [22, 24, 40] due to their powerful feature representation capability. With the lack of labels, most of the unsupervised deep hashing methods construct a pairwise similarity matrix with pre-trained deep features. For example, Yang et al. [49] compute the similarity matrix based on the observation that the distribution of the cosine distance for point pairs can be estimated by two half Gaussian distributions. Once the similarity matrix is built, in most of the existing methods, it will be fixed to guide the hash code learning process. However, such a similarity matrix may be unreliable because it is computed with the pre-trained deep features and ignores the semantic information in the downstream retrieval task. To tackle this issue, Shen et al. [35] employ a straightway by reconstructing a similarity graph with the fine-tuning features through the Gaussian kernel. However, such specific construction schemes mainly focus on the local structure, which limits the model performance, and also, the reconstruction strategy suffers from high computational costs.

In addition, most of the existing methods [2, 7, 47] ignore the semantic importance of data pairs during their design of the similarity-preserving loss function. As such data pairs are treated equivalently in the learning process. A few of works [30, 49, 51, 52] primitively divide the data pairs into different types like confident or unconfident, they still fail to provide a fine-grained measurement of the importance of different data pairs. The pairs in the same class are

still treated equivalently. However, the priority of different data pairs should be different. Intuitively, the dissimilar data pairs might contain more discriminative information, and be more informative for model training. Otherwise, the informative data pairs will be buried by large number of less-informative training pairs and decline hash learning.

To deal with the aforementioned issues, we propose a novel **Deep Self-Adaptive Hashing (DSAH)** method to adaptively explore the semantic information under training, and provide two innovative components. The pipeline is shown in Figure 1. In particular, to overcome the disadvantage of fixed pre-computed similarity, we adopt an **Adaptive Neighbor Discovery (AND)** technique, which mines underlying neighbor relationships behind the fine-tuning features gradually during training, and update the initial similarity graph continuously. Further, we introduce an adjustable pairwise weight term **Pairwise Information Content (PIC)** to distinguish the different importance of data pairs. In this way, the data pairs with more information would gain greater weights and contribute more to model learning. These two techniques enable DSAH to fully explore the semantic information behind the data pairs and learn better hash codes in a self-adaptive manner. The main contributions of this paper can be summarized as follows:

- We propose a novel **DSAH** method to yield better hash codes in a self-adaptive manner with the full exploration of semantic information behind the data pairs.
- We propose the **AND** technique to refine the pre-computed similarity matrix with the fine-tuned representations during the learning process, which can adaptively capture the implicit neighbor relationships and improve the hash learning performance.
- We propose the **PIC** to measure the importance of different data pairs and use it to weigh them during training, where more dissimilar data pairs will be assigned larger weights to augment their discriminative power. These weights are further adaptively updated with the pairwise similarity distribution when the training iterates.
- The extensive experiments on several benchmark datasets show that our DSAH is indeed effective and achieves superior performance, which achieves 6.2%, 8.7%, 2.7% improvement over the best baseline on CIFRA-10, FLICKR25K, and NUS-WIDE datasets, respectively.

2 NOTATION AND PROBLEM DEFINITION

Let us introduce some notations for this paper. We use boldface uppercase letters $\mathbf{A}$ to represent a matrix, in which $\mathbf{a}_i$ represents the i-th row of $\mathbf{A}$, a_{ij} represents the element of $\mathbf{A}$ which is in the i-th row and the j-th column. $\|\cdot\|$ denotes the L2 norm and $\|\cdot\|_F$ represents the Frobenius norm. $\mathbf{1}(\cdot)$ is the indicator function. $\tanh(\cdot)$ is the hyperbolic tangent function. $\text{sgn}(\cdot)$ is the sign function, which outputs +1 for positive numbers, or −1 otherwise.

$$\text{sgn}(x) = \begin{cases} 1, & x \geq 0 \\ -1, & x < 0 \end{cases} \quad (1)$$

Given a dataset contain n samples $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\} \in \mathbb{R}^{d \times n}$ without human annotations, where d is the dimension of samples. The goal of hashing is to learn a function $\mathcal{H} : \mathbf{x}_i \rightarrow \mathbf{b}_i$

that maps the dataset $\mathbf{X}$ into a set of compact binary hash codes $\mathbf{B} = \{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n\} \in \{+1, -1\}^{l \times n}$ where l is the length of codes.

3 RELATED WORK

In this section, we briefly review the traditional shallow hashing methods and recent deep hashing methods. These methods can be divided into supervised and unsupervised categories, and we mainly focus on unsupervised hashing methods.

3.1 Shallow Hashing

Generally, traditional shallow hashing methods can be classified into two categories: data-independent hashing methods and data-dependent methods. Data-independent hashing methods [1, 9, 20] typically use random projection to generate binary code. Locally Sensitive Hashing (LSH) [1] is a representative data-independent hashing method, which enables similar data to share similar hash codes in Hamming space. However, data-independent hashing methods require longer binary codes to achieve higher accuracy, which also results in higher storage costs. The data-dependent hashing methods [6, 10, 25, 46] have also received increasing attention in recent years, which aim to learn compact and high-quality binary codes from specific data. One of the most classical data-dependent hashing methods is Spectral Hashing (SH) [46], which was developed to learn binary codes by preserving local manifold structures. The objective function of SH can be written as:

$$\min_{\mathbf{B}} \sum_{ij} w_{ij} \|\mathbf{b}_i - \mathbf{b}_j\|^2$$
$$s.t. \mathbf{b}_i \in \{-1, +1\}^l, \quad \sum_i \mathbf{b}_i = 0, \quad \frac{1}{n} \sum_i \mathbf{b}_i \mathbf{b}_i^T = I \quad (2)$$

where $w_{ij} = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2/\varepsilon)$ is the similarity weight between x_i and x_j. ε is the balance parameter. The optimization of Eq. 2 is equivalent to balanced graph partitioning and is an NP-hard problem. SH solves this problem by introducing spectral relaxation, which removes the discrete constraint. The objective function of SH is to binarize the input so that the Hamming distance approximates the Euclidean distance. Generally, the shallow hashing methods usually encode the input images with hand-crafted features (e.g, $\|\mathbf{x}_i - \mathbf{x}_j\|^2$ of w_{ij}), which is insufficient to characterize the semantic information of data and thus degrade the performance.

3.2 Deep Hashing

Recently, deep hashing methods [3, 8, 21, 23, 26, 31, 50] show promising performance compared with the shallow hashing methods. Xia et al. [47] is one of the earliest works that introduces the deep neural networks into hashing function learning, where the hash codes are learned by minimizing the following loss function

$$\mathcal{L}_0 = \sum_{i=1}^{n} \sum_{j=1}^{n} \left(s_{ij} - w_{ij} \right)^2 \quad (3)$$

where $s_{ij} = \frac{1}{l} \mathbf{z}_i^T \mathbf{z}_j$ is the inner product between deep features. $\mathbf{z}_i \in [-1, +1]^l$ is the relaxation of hash codes $\mathbf{b}_i$ since it is difficult to directly optimize the discrete variables. Suggested by Cao et al. [2], cosine similarity might be a better choices to mitigate the diversity
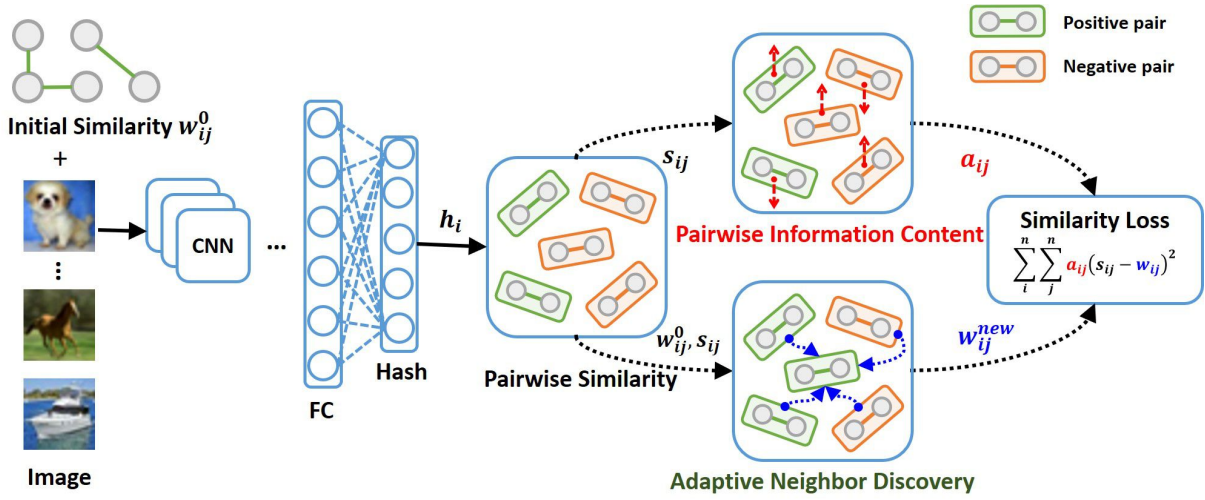
Figure 1: The pipeline of the proposed Deep Self-Adaptive Hashing (DSAH). First, we initialize a similarity matrix based on the original features and extract the image features via VGG-19 to compute the pairwise similarity. Next, we provide two novel technologies: (i) We introduce PIC (Upper branch) to augment the priority of informative data pairs. (ii) We propose AND (Down branch) to refine the initialized similarity matrix and mine the implicit neighbors' relationships with learned representation. These two components assist DSAH to learn hash codes in a self-adaptive manner.

of code lengths and improve the performance, which defined as:

$$s_{ij} = cos(\mathbf{z}_i, \mathbf{z}_j) = \frac{\mathbf{z}_i^T \mathbf{z}_j}{\|\mathbf{z}_i\|\|\mathbf{z}_j\|} \quad (4)$$

where w_{ij} of Eq.3 is the binary pairwise similarity, which is equal to +1 if $\mathbf{x}_i$ and $\mathbf{x}_j$ are semantically similar, or −1 otherwise.

In supervised cases, w_{ij} is constructed according to labels. But in the unsupervised settings, the labels are unavailable. To tackle this problem, Yang et al. [49] empirically study the deep feature statistics and estimate two half Gaussian distributions to calculate the similarity. Song et al. [38] employs the k Nearest Neighborhoods (k-NN) to compute a binary similarity with pre-trained features. While Zhang et al. [52] construct a hybrid-similarity matrix in advance. Tu et al. [43] utilizes the local manifold structure to construct the similarity graph. However, in these methods, the semantic similarity matrix is pre-computed on the original feature space and is separated from the hash learning process. Intuitively, a good similarity representation is beneficial to hashing function learning, and also, hashing function might provide meaningful feedback to similarity representation. Therefore we hope that the similarity matrix and hashing function can be learned simultaneously, which inspires us to design an adaptive similarity updating method rather than reconstructing a similarity structure based on the updated features [35]. Moreover, the existing similarity-preservation loss function (e.g, Eq.3) usually neglects the priorities between different data pairs in the learning process. Though several works [3, 30, 49, 52] divide the data pairs into different groups based on their pairwise similarity, these partitions are also pre-defined and fail to capture the fine-grained priority between data pairs since the pairs within the same group are still treated fairly. Thus the informative data pairs may be buried in a large number of uninformative training pairs.

4 METHODOLOGY

In this section, we develop our Deep Self-Adaptive Hashing model. The overall architecture of DSAH is illustrated in Figure 1, which contains three modules: Feature Extraction, Adaptive Neighbor Discovery and Pairwise Information Content. We will demonstrate each module in the following section in detail.

4.1 Feature Extraction

We apply the VGG-19 [37] for hash function learning and denote it as $\mathcal{F}(\cdot, \Theta)$ with network parameters Θ. VGG-19 contains five convolutional layers and three fully-connected layers, to make the network suitable for hash learning, we replace the last layers with a fully-connected layer with 1000 hidden units and followed by a hash layer, in which the number of neurons is equal to hash codes length l. To resolve the ill-posed gradient of $sgn(\cdot)$, we adopt the $tanh(\cdot)$ as the activation function of the hash layer and then we get the approximation of hash code $\mathbf{b}_i$ as follows:

$$\mathbf{z}_i = \tanh(\mathcal{F}(\mathbf{x}_i; \Theta)) \in [-1, +1]^l \quad (5)$$

Once we finish the training process, we can obtain the discrete hash code $\mathbf{b}_i$ as follows:

$$\mathbf{b}_i = sgn(\mathcal{F}(\mathbf{x}_i; \Theta)) \in \{-1, +1\}^l \quad (6)$$

4.2 Adaptive Neighbor Discovery

In this subsection, we propose the AND, which consists of two steps: similarity matrix initialization and updating.

4.2.1 Initialization. Recent works [38, 43, 49, 52] have shown that the rich semantic structure can be captured via an elaborately designed similarity matrix. In our AND, we construct an initial similarity matrix at the beginning based on k-NN.

We first extract deep features from the relu-7 layer of a pre-trained VGG-19 model and then select k_1 images with the highest cosine similarity as the neighbors of each image. Then we construct a *Low-order* similarity matrix $\mathbf{W}_L$ as follows:

$$(\mathbf{W}_L)_{ij} = \begin{cases} +1, & \text{if } \mathbf{x}_j \text{ is } k_1\text{-NN of } \mathbf{x}_i, \\ -1, & \text{otherwise.} \end{cases} \tag{7}$$

Notably, we consider that if the neighbors of two images are highly similar, then these two images should also be very similar. Thus we calculate the similarity of two images neighbors using the expression $\frac{1}{1+\|(\mathbf{W}_L)_i - (\mathbf{W}_L)_j\|}$. Then we select the top k_2 samples to construct a *High-order* similarity matrix $\mathbf{W}_H$ as:

$$(\mathbf{W}_H)_{ij} = \begin{cases} +1, & \text{if } \mathbf{x}_j \text{ is } k_2\text{-NN of } \mathbf{x}_i, \\ -1, & \text{otherwise.} \end{cases} \tag{8}$$

In order to take full advantage of these two similarity matrices, we define our initial similarity matrix $\mathbf{W}^0$ by combining $\mathbf{W}_L$ and $\mathbf{W}_H$ together, which is based on the assumption that two similar images should not only be similar in feature space but also share similar neighbors.

$$(\mathbf{W}^0)_{ij} = \begin{cases} +1, & \text{if } (\mathbf{W}_L)_{ij} = +1 \text{ and } (\mathbf{W}_H)_{ij} = +1 \\ -1, & \text{otherwise.} \end{cases} \tag{9}$$

However, a drawback of $\mathbf{W}^0$ is that it is built with the pre-trained features, which might contain noisy data. Ideally, we hope that the similarity maintains consistency with the fine-tuning features, while the reconstruction is time-consuming. Hence, we next propose an efficient updating strategy to refine the $\mathbf{W}^0$.

4.2.2 Updating. AND provides a progressive mechanism for similarity updating. Given a similarity matrix $\mathbf{W}^r$ in the r-th round (or $\mathbf{W}^0$ in beginning), we first use it to update the model parameters Θ. (e.g, Optimizing the $\mathcal{L}_0$ in Eq. 3). Next, we measure the pairwise cosine similarity $\mathbf{S}^r = \{s_{ij}^r\} \in \mathbb{R}^{n \times n}$ over the training data pairs through Eq.4 and estimate a threshold m^r by

$$m^r = \mu^r + \gamma \cdot \sigma^r \tag{10}$$

where the μ^r and σ^r denote the mean and standard derivation of cosine similarity s_{ij}^r for those data pairs with $w_{ij}^r = +1$, which is defined as:

$$\begin{cases} \mu^r = \dfrac{\sum_{i=1}^n \sum_{j=1}^n s_{ij}^r \cdot \mathbf{1}(w_{ij}^r = +1)}{n_+} \\[2mm] \sigma^r = \left(\dfrac{\sum_{i=1}^n \sum_{j=1}^n \left(s_{ij}^r - \mu^r\right)^2 \cdot \mathbf{1}(w_{ij}^r = +1)}{n_+} \right)^{\frac{1}{2}} \end{cases} \tag{11}$$

where n_+ is equal to $\sum_{i=1}^n \sum_{j=1}^n \mathbf{1}(w_{ij}^r = 1)$, which counts the number of neighbors in $\mathbf{W}^r$. γ is a hyper-parameter that control the threshold. Finally, we renew the $\mathbf{W}^r$ as follows:

$$w_{ij}^{r+1} = \begin{cases} +1, & \text{if } w_{ij}^r = -1 \text{ and } s_{ij}^r \geq m^r, \\ w_{ij}^r, & \text{otherwise.} \end{cases} \tag{12}$$

Analysis. The AND is motivated by the prediction interval, aiming to design a dynamic criterion Eq.10 based on the global distribution of learned features. The tuition behind Eq.12 is that

these dissimilar data pairs with similarity s_{ij} higher than average similarity μ of similar pair sets could probably be treated as candidate similar data pairs in next epoch, where γ controls the tolerance, a larger γ is more serious while a smaller γ is looser. The reason why we focus on similar pairs instead of dissimilar pairs is that the neighborhoods-based $\mathbf{W}^0$ contains only a few similar pairs ($n_+ \ll n^2$). There is still a lot of information (e.g, neighbors) worth mining in a large number of dissimilar pairs, as the model capacity increases, this information would be better distinguished. Therefore, AND is able to adaptively adjust the similarity $\mathbf{W}^0$ according to the learned representation, in a progressive mechanism.

4.3 Pairwise Information Content

In PIC, we discuss the priority of different data pairs for model training. Though Eq. 3 provides a scheme to learn hash codes from similarity matrix $\mathbf{W}$, it neglects the importance of different data pairs. All the data pairs with their similarities are treated fairly when calculating loss, so that some informative data pairs may be buried in a large number of samples. To tackle this issue, we propose the PIC, which assigns an adaptive weight for each data pair. If a data pair is more important, it will receive a larger weight and contribute more to hashing function learning. So, we define the following loss function:

$$\mathcal{L}_1 = \sum_{i=1}^n \sum_{j=1}^n a_{ij} \left(s_{ij} - w_{ij}\right)^2 \tag{13}$$

where a_{ij} is the weight for the data pair (i, j) which represents the importance of this data pair.

According to the information theory [33], given an event Φ with probability $p(\Phi)$, its *information content* is defined as the negative log-likelihood:

$$I(\Phi) = -\log p(\Phi) \tag{14}$$

If an event Φ has a probability 1 of occurring, then its information content is $-\log(1) = 0$ and yields no information. While an event with probability 0, its information content is $+\infty$. Inspired by Eq. 14, we design the following definition:

Definition 4.1. Given a data pair (i, j), we define an event Φ_{ij} with probability p_{ij}. The Φ_{ij} indicates that the image j is the top-1 retrieval result of the given query image i, which is an optimum matching of image retrieval. Based on this, we define the **pairwise information content** a_{ij} as:

$$a_{ij} = I(\Phi_{ij}) = -\log(p_{ij}) \tag{15}$$

Typically, if an image j is the top-1 retrieval result of the query image i, then i and j should be the most similar. Therefore probability p_{ij} can be expressed as a measure of relative similarity between image i and image j, and we define p_{ij} as

$$p_{ij} = \frac{\exp(s_{ij}/\tau)}{\sum_{g=1}^n \sum_{k=1}^n \exp(s_{gk}/\tau)} \tag{16}$$

where τ is a temperature parameter and s_{ij} is the pairwise cosine similarity in Eq. 4.

Analysis. Given a data pair (i, j), if p_{ij} is higher, then the information content $I(\Phi_{ij})$ that the image j can be retrieved through the image i is lower. (A special case is to use an image to retrieve itself, then $I(\Phi_{ij})$ should be lowest or close to 0.) On the contrary,
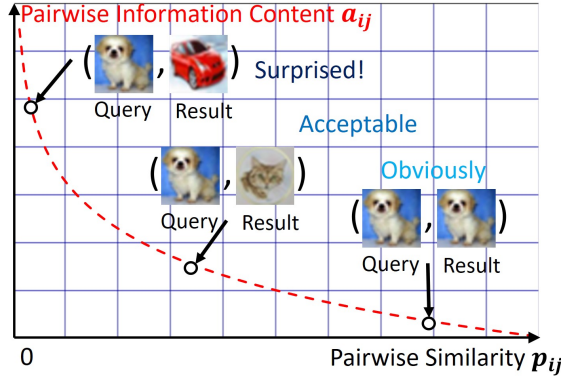
Figure 2: The relationship between PIC weight a_{ij} and relative pairwise similarity p_{ij}. If a data pair is highly dissimilar, their a_{ij} will be higher, otherwise, it will be smaller.

retrieving an image via a highly dissimilar query would bring a lot of information. Figure.2 shows an example, given a query image of *Dog*, if we retrieve a *Car*, we would be more surprised than retrieving a *Cat*. So that we have $I(\Phi_{Dog,Car}) > I(\Phi_{Dog,Cat})$ while $sim(Dog, Car) < sim(Dog, Cat)$, where sim denotes the pairwise similarity. The tuition behind PIC can work is that a data pair contains highly dissimilar images would provide richer distinctive information, so it should be given a larger weight contributing to the hashing learning. The PIC is based on the pairwise similarity distribution p_{ij}, it is adjustable and adaptive. Moreover, our PIC could also be expressed as a kind of pairwise attention mechanism. Particularly, when a_{ij} is equal to constant 1, the importance of data pairs will be the same and the Eq. 13 will degenerate to Eq. 3.

4.4 Objective Function

In addition, to guarantee the quality of learned hash codes, we introduce a loss $\mathcal{L}_2 = \|\mathbf{Z} - \mathbf{B}\|_F^2$ to minimize the quantization error between the variables $\mathbf{Z}$ and the binary codes $\mathbf{B}$. Therefore, the total loss function can be formulated as

$$\mathcal{L} = \mathcal{L}_1 + \lambda \mathcal{L}_2$$
$$= \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \left(s_{ij} - w_{ij} \right)^2 + \lambda \|\mathbf{Z} - \mathbf{B}\|_F^2 \quad (17)$$
$$= \sum_{i=1}^{n} \sum_{j=1}^{n} -\log(p_{ij}) \left(\frac{\mathbf{z}_i^T \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|} - w_{ij} \right)^2 + \lambda \|\mathbf{Z} - \mathbf{B}\|_F^2$$

4.5 Optimization

Our DSAH consists of two main parts and since the AND is not end-to-end, we separate the optimization process into two alternate steps. First, we fix the similarity matrix $\mathbf{W}$ and optimize the network parameter Θ by back propagation (BP) with a mini-batch sampling.

$$\Theta \leftarrow \Theta - \eta \nabla_\Theta(\mathcal{L}) \quad (18)$$

where η is the learning rate and ∇_Θ represents a derivative of Θ.

Second, we fix the Θ and update the similarity matrix $\mathbf{W}$ according to Eq. 12. Once the model training is completed, we can obtain the discrete hash code $\mathbf{B} \in \{-1, +1\}^{n \times l}$ through Eq. 6. The detailed algorithm of our proposed DSAH is described in Algorithm 1.

Algorithm 1 Deep Self-Adaptive Hashing (DSAH)

Require: Image set $\mathbf{X}$, hash code length l, temperature τ, weight coefficient λ, factor γ, rounds R, epochs per round T, learning rate η
1: Initialize the network parameters $\mathcal{F}(\cdot, \Theta)$.
2: Initialize the semantic similarity matrices $\mathbf{W}^0 \in \{+1, -1\}^{n \times n}$ by Eq. 9.
3: **for** $r = 1$ to R **do**
4:      // *Hash Function Learning with PIC*
5:      **for** $t = 1$ to T **do**
6:          Update the $\mathbf{Z}$ via Eq. 5.
7:          Update the $\mathbf{S}$ via Eq. 4.
8:          Update the $\mathbf{P}$ via Eq. 16.
9:          Update the Θ via Eq. 18.
10:      **end for**
11:      // *Similarity Updating by AND*
12:      Update the $\mathbf{W}^r$ via Eq. 12.
13: **end for**
14: Obtain the $\mathbf{B}$ via Eq. 6.
Ensure: Hash codes set $\mathbf{B}$

---

5 EXPERIMENTS

In this section, we conduct experiments on several public benchmark datasets and evaluate our DSAH method. Particularly, we design the experiments to study the following questions:

- **Q1**: Compared with the state-of-the-art unsupervised hashing algorithms, does our DSAH method outperform them?
- **Q2**: How do the components: PIC and AND, affect the performance of DSAH separately? How do they work?
- **Q3**: Is DSAH sensitive to hyper-parameters?
- **Q4**: How about the efficiency of DSAH?
- **Q5**: What is the qualitative result of DSAH?

5.1 Datasets

Following most hashing-based retrieval methods [7, 30, 38], we adopt the following three widely used public benchmark datasets to evaluate the model performance: CIFAR-10, FLICKR25K and NUS-WIDE. The basic information is listed in Table.2, while the setting details be introduced as follows:

Dataset	CIFAR-10	FLICKR25K	NUS-WIDE
Multi-Label	×	✓	✓
# Images	60,000	25,000	269,648
# Classes	10	24	21
# Train Set	10,000	10,000	10,500
# Query Set	1,000	1,000	2,100
# Retrieval Set	59,000	24,000	267,548

Table 2: Characteristics of evaluation datasets.

CIFAR-10 [19]: Followed the setting of [38], we randomly selected 100 images for each class as the query set, 1,000 in total. Then we used the remaining images as the retrieval set, among them, we randomly selected 1,000 images per class as the training set.

FLICKR25K [14]: We randomly selected 1000 images as query set and the remaining images were left for retrieval set. In the retrieval set, we randomly chose 10,000 images as the training set.

Method	Reference	CIFAR-10				FLICKR25K				NUS-WIDE			
		16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
LSH+VGG [1]	STOC-02	0.177	0.192	0.261	0.304	0.596	0.619	0.650	0.666	0.385	0.455	0.446	0.567
SH+VGG [46]	NeurIPS-09	0.254	0.248	0.229	0.293	0.661	0.608	0.606	0.614	0.508	0.449	0.441	0.505
ITQ+VGG [10]	PAMI-13	0.269	0.295	0.316	0.350	0.709	0.696	0.684	0.720	0.519	0.576	0.598	0.651
AGH+VGG [25]	ICML-11	0.397	0.428	0.441	0.435	0.744	0.735	0.771	0.703	0.563	0.698	0.725	0.722
SP+VGG [48]	CVPR-15	0.280	0.343	0.365	0.406	0.726	0.705	0.713	0.703	0.581	0.603	0.673	0.681
SGH+VGG [6]	ICML-17	0.286	0.320	0.347	0.395	0.608	0.657	0.693	0.689	0.463	0.588	0.638	0.670
GH [39]	NeurIPS-18	0.355	0.424	0.419	0.416	0.702	0.732	0.753	0.760	0.599	0.657	0.695	0.712
SSDH [49]	IJCAI-18	0.241	0.239	0.256	0.246	0.710	0.696	0.737	0.760	0.542	0.629	0.635	0.606
BGAN [38]	AAAI-18	0.535	0.575	0.587	0.591	0.766	0.770	0.795	0.802	0.719	0.745	0.761	0.759
MLS^{3}RDUH [43]	IJCAI-20	0.562	0.588	0.595	0.582	0.797	0.809	0.809	0.804	0.730	0.754	0.764	0.769
TBH [36]	CVPR-20	0.432	0.459	0.455	0.474	0.779	0.794	0.797	0.799	0.678	0.717	0.729	0.739
DSAH	**Proposed**	**0.596**	**0.617**	**0.622**	**0.635**	**0.805**	**0.816**	**0.831**	**0.836**	**0.747**	**0.769**	**0.787**	**0.793**

Table 1: MAP@5000 results on CIFAR10, FLICKR25K and NUS-WIDE. The best result is shown in boldface.

NUS-WIDE [5]: According to the setting in [53], we selected 21 most frequent classes from the dataset and each class contains at least 5,000 related images. We randomly selected 2100 images as the query set and the remaining images were used as a retrieval set. We also randomly selected 10,500 images for training.

For the latter two multi-label datasets, if the retrieved image shares at least one label with the query image, it is considered to be associated with the query image.

5.2 Experiment Setup

Metrics: Similar to [35, 45, 49], we employed four widely used evaluation metrics to evaluate the retrieval performance, including Mean Average Precision (**MAP**), Precision of the top N retrieved images (**Precision@N**), Precision curve (**Precision Curve**), and Precision-Recall curves (**PR Curve**). For a fair comparison, all the methods used the same training and query sets.

Baseline methods: We compared our method with eleven unsupervised hashing methods, including six shallow hashing methods: **LSH** (Andoni and Indyk [1]), **SH** (Weiss et al. [46]), **ITQ** (Gong et al. [10]), **AGH** (Liu et al. [25]), **SP** (Xia et al. [48]), **SGH** (Dai et al. [6]) and five deep hashing methods: **GH** (Su et al. [39]), **SSDH** (Yang et al. [49]), **BGAN** (Song et al. [38]), **MLS^{3}RDUH** (Tu et al. [43]) and **TBH** (Shen et al. [36]). The parameters and architectures of the compared methods were according to the setting provided by the original papers. For a fair comparison, all shallow hashing methods used 4096-dimensional features generated by the relu7 layer of VGG19 [37] pre-trained on ImageNet, as same as the deep features used in the five deep hashing methods during their similarity structures construction.

Implementation details: Our DSAH is implemented based on the Tensorflow framework, while all the experiments are conducted on a workstation with an Intel l5-8500 CPU, and an Nvidia GTX2080 GPU. In the initial similarity construction (Sec 4.2.1), the k_1 and k_2 are set to the same value 500. When training the network, each training image was resized to 224×224 as input. We set λ equal to 10 and adopt the adam optimization with learning rate η equal to 1e-4, and the batch size was set to 50. The τ is set to 1, while the γ is set to {1, 0, 1} for CIFAR-10, FLICKR25K and NUS-WIDE datasets respectively. And the R and T are set to 3 and 10 respectively.

5.3 Comparison Results and Discussions (Q1)

5.3.1 MAP & Precision. The performance of our DSAH and baseline methods in terms of MAP@5000 are shown in Table. 1. We can observe that our proposed method significantly outperforms other methods on the three datasets. On CIFAR-10, DSAH obtains an increase of 6.1% (16 bits), 4.9% (32 bits), 4.5% (64 bits), 9.1% (128 bits) comparing with the best competitor MLS^{3}RDUH respectively. On FLICKR-25K and NUS-WIDE, DSAH achieves average 8.7% and 2.7% improvement compared to the MLS^{3}RDUH.

In the practical scenarios, we pay more attention to the top retrieval responses. Thus, we further evaluate the performance on the precision of the top-100 for each method. The results are demonstrated in Table. 3, where we display the result of the shortest (16 bits) and longest (128 bits) binary codes. On FLICKR25K, DSAH achieves 1.5% (16 bits) and 2.3% (128 bits) increases comparing with TBH. On NUS-WIDE, DSAH is improved by 1.7% (16 bits) and 2.9% (128 bits) compared to BGAN. Therefore, both the Table.1 and Table.3 demonstrate that DSAH achieves better performance compared to previous unsupervised hashing methods.

Dataset	CIFAR-10		FLICKR25K		NUS-WIDE	
Methods	16 bits	128 bits	16 bits	128 bits	16 bits	128 bits
LSH+VGG [1]	0.175	0.411	0.608	0.745	0.393	0.651
SH+VGG [46]	0.291	0.446	0.715	0.700	0.518	0.625
ITQ+VGG [10]	0.283	0.458	0.760	0.818	0.545	0.739
AGH+VGG [25]	0.444	0.617	0.794	0.877	0.572	0.794
SP+VGG [48]	0.284	0.514	0.788	0.852	0.601	0.756
SGH+VGG [6]	0.285	0.536	0.622	0.814	0.457	0.764
GH [39]	0.410	0.519	0.773	0.843	0.651	0.777
SSDH [49]	0.216	0.269	0.753	0.800	0.595	0.662
BGAN [38]	0.591	0.691	0.839	0.884	0.749	0.794
MLS^{3}RDUH [43]	**0.618**	0.677	0.851	0.868	0.748	0.791
TBH [36]	0.499	0.586	0.849	0.877	0.702	0.775
DSAH	0.616	**0.691**	**0.862**	**0.897**	**0.762**	**0.817**

Table 3: Precision@100 results on CIFAR10, FLICKR25K and NUS-WIDE. The best result is shown in boldface.

5.3.2 Precision Curve & PR Curve. To further illustrate the effectiveness of DSAH, we display the Precision curve and PR curve of 64 and 128 bits in Figure. 3. The PR Curve Figure.3 (a)(c)(e)(g)(i)(k) clearly displays the precision at different recall values, which is a good representation of overall performance. In general, a larger
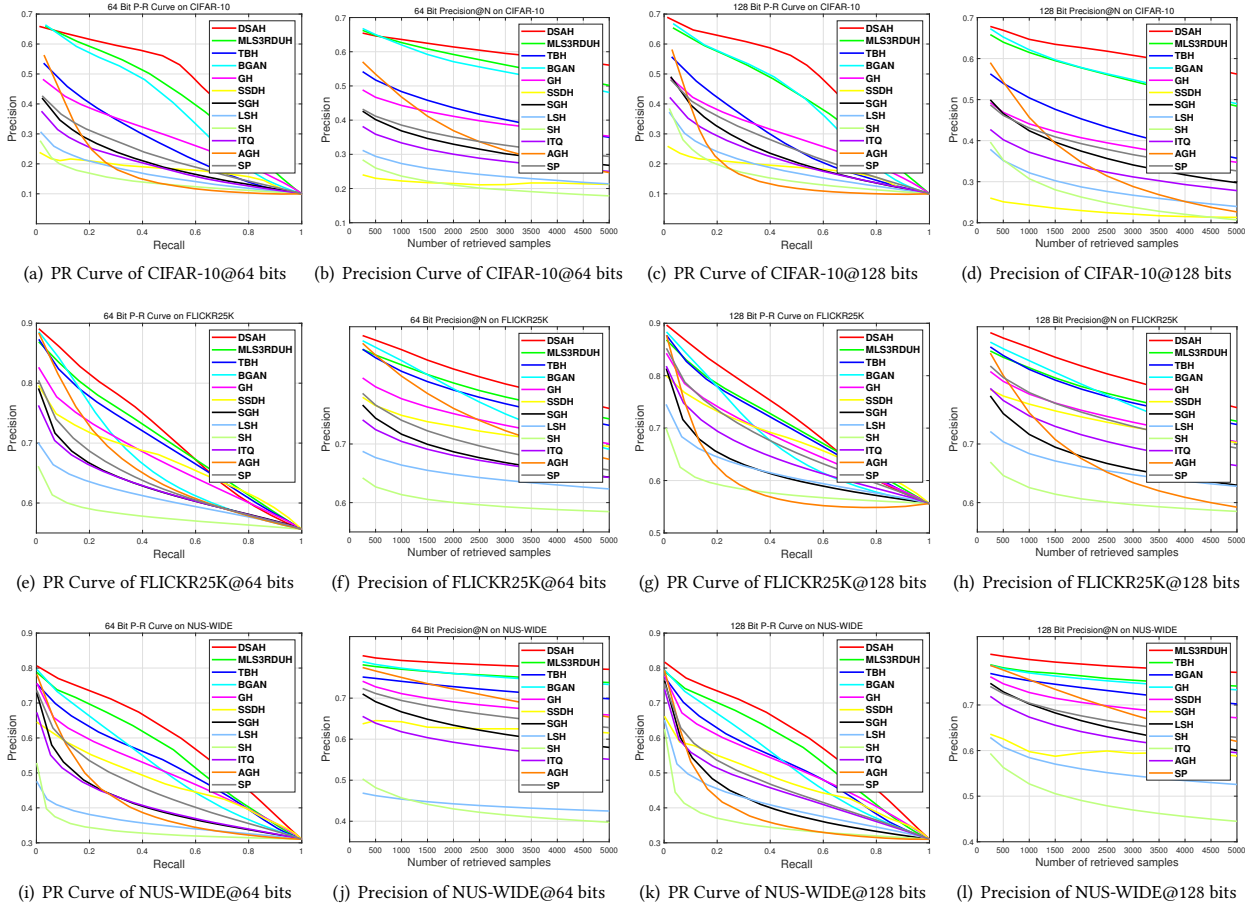
(a) PR Curve of CIFAR-10@64 bits (b) Precision Curve of CIFAR-10@64 bits (c) PR Curve of CIFAR-10@128 bits (d) Precision of CIFAR-10@128 bits

(e) PR Curve of FLICKR25K@64 bits (f) Precision of FLICKR25K@64 bits (g) PR Curve of FLICKR25K@128 bits (h) Precision of FLICKR25K@128 bits

(i) PR Curve of NUS-WIDE@64 bits (j) Precision of NUS-WIDE@64 bits (k) PR Curve of NUS-WIDE@128 bits (l) Precision of NUS-WIDE@128 bits

Figure 3: Precision-Recall (PR) curves and Precision@N curves on the three datasets for 64 and 128 bits length.

area under the PR curve indicates better performance. It can be seen that our PR curve covers more areas in most cases, which means that when the precision is equal, the proposed method will recall more related images; when the number of recall related images is equal, the proposed method has higher precision. Thus, DSAH yields a stable and superior performance. Similar to the PR Curve setting, we display the Precision Curves in Figure.3 (b)(d)(f)(h)(j)(l). It can be seen that the precisions of our methods is relatively higher than the precisions of other methods in most cases, which indicates that given a fixed number of retrieval samples, our method can obtain higher precision outperform other methods.

5.4 Ablation Study (Q2)

Since our method consists of two major components: PIC and AND, we further verify their effectiveness.

5.4.1 Effect of PIC. In order to validate the efficiency of PIC, we design the following variants with different a_{ij} in Eq. 13:

- **PIC^0**: a_{ij} is equal to constant 1, and it would degenerate to Eq. 3. This could be treated as a baseline.
- **PIC**: a_{ij} is equal to $-\log(p_{ij})$. This is our proposed PIC.

- **PIC$^-$**: a_{ij} is equal to $-\log(1 - p_{ij})$. This could be regarded as an opposite version of our PIC.

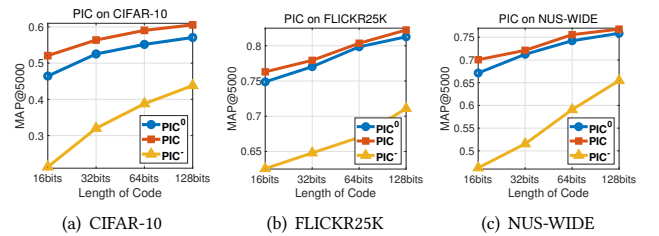(a) CIFAR-10 (b) FLICKR25K (c) NUS-WIDE

Figure 4: Effect of PIC on three datasets with MAP@5000, the code lengths varying from 16 to 128.

We report the experimental results on the three datasets in Figure 4, which are evaluated with MAP@5000. As shown in Figure. 4 (a)-(c), we could observe that: Firstly, compared with baseline PIC^0, introducing PIC weights is able to obtain average 8.2%, 1.2%, 2.1% MAP improvements on CIFAR-10, FLICKR25K and NUS-WIDE datasets,

respectively. These results indicate that PIC indeed improves the discriminative power of our model. Secondly, the performance degradation of the third PIC⁻ shows that assigning large weights to similar data pairs will degenerate the performance, which indicates that dissimilar data pairs should receive more attention rather than similar data pairs. These results validate the effectiveness of our proposed weighting method PIC.

How PIC works. To further understand PIC, we display the heatmap of pairwise similarity s_{ij} and their PIC weights a_{ij} within a batch size in Figure. 5. It is shown that those data pairs with higher similarity, especially in the diagonal (self-similar), would receive a lower weight after PIC assignment and dissimilar pairs will reach higher weights.
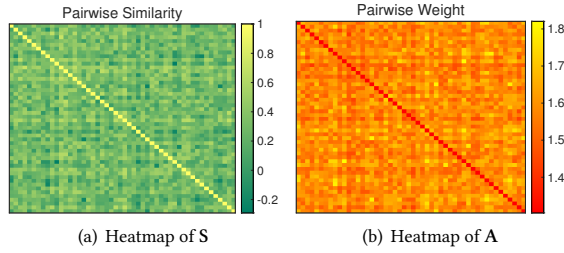

(a) Heatmap of **S**   (b) Heatmap of **A**

Figure 5: The heatmap of (a) pairwise similarity s_{ij} and (b) corresponding PIC weight a_{ij} within a batch size.

5.4.2 Effect of AND. To reveal the effectiveness of AND, we conduct an experiment to compare the following variants:

- **AND⁰**: Training via Eq. 3 with $\mathbf{W}^0$, which is baseline.
- **AND**: Training via Eq. 3 with updated $\mathbf{W}^r$ via AND.
- **AND+PIC**: Training via Eq. 13 with updated $\mathbf{W}^r$ via AND.

We report the results evaluated with MAP@5000 on the three datasets in Figure. 6. By employing the AND, the baseline model can be generally improved, achieving 1.7%, 1.8%, 1.0% average improvements on three datasets, respectively. Furthermore, the performance can be greatly improved by introducing both PIC and AND, resulting in 8.9%, 3.4% and 3.8% average improvements over baseline in three datasets. The above experiments reveal the effectiveness of PIC and AND of DSAH.
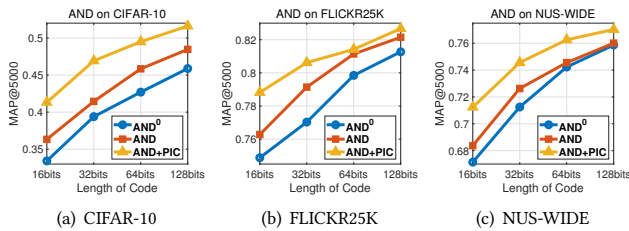

(a) CIFAR-10   (b) FLICKR25K   (c) NUS-WIDE

Figure 6: Effect of AND on three datasets with MAP@5000, the code length varying from 16 to 128.

How AND works. To better understand the mechanism of AND, we enlarge the round R to 10 and record the changing of threshold m

and the number of neighbors pairs n_+. Also, to monitor the quality of the updated similarity matrix $\mathbf{W}^r$, we compute an indicator matrix $\mathbf{G} \in \{-1, +1\}^{n \times n}$, which element g_{ij} is equal to +1 iff data pairs (i, j) belong to same class, otherwise equal to -1. Next, we treat AND as a binary classification problem and $\mathbf{G}$ as ground-truth, and then design a metric F_w to measure the quality of updated matrix $\mathbf{W}^r$, which is inspired by F-score:

$$F_w = \frac{2}{\alpha^{-1} + \beta^{-1}}$$

where α is equal to $\frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{1}(w_{ij}^t = g_{ij})}{\sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{1}(w_{ij}^t = +1)}$, denotes the precision of updated similarity matrix $\mathbf{W}^r$. And β denotes the recall of $\mathbf{W}^r$, defined as $\frac{\sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{1}(w_{ij}^t = g_{ij})}{\sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{1}(g_{ij} = +1)}$. In Figure. 7. We could see that in each round, additional data pairs are added to the neighbors set, and the number of similar data pairs converges eventually. This is mainly controlled by the increase of the adjustable threshold m. Notably, the climbing F_w confirms the effect of AND, which indeed refines the quality of the similarity matrix and leads to the model's improvement, especially in the first updating.
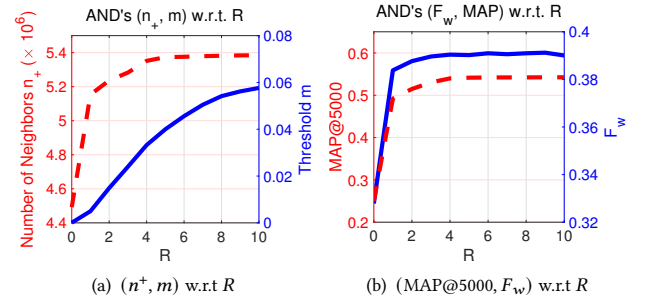

(a) (n^+, m) w.r.t R (b) (MAP@5000, F_w) w.r.t R

Figure 7: The Mechanism of AND. We monitor the AND's (n_+, m, MAP, F_w) w.r.t round R.

5.5 Parameter Sensitivity (Q3)

5.5.1 Study of λ. In Figure. 8 (a), we study the influence of the quantization effect with different λ over three datasets, where the code length is 64. From this figure, the performance will degrade when λ is larger than 10. And the recommended value for λ is 10, which would bring 4.3%, 3.1% and 2.8% MAP improvements in CIFAR-10, FLICKR25K and NUS-WIDE, respectively.
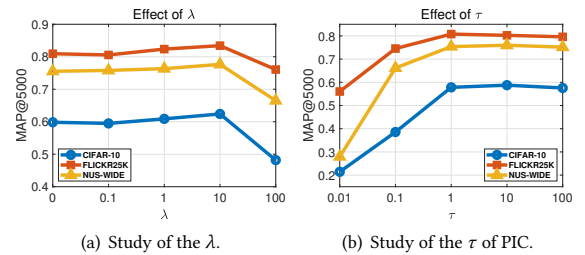

(a) Study of the λ.   (b) Study of the τ of PIC.

Figure 8: MAP@5000 under (a) different λ and (b) different τ of PIC, the code length is 64.

5.5.2 Study of PIC. In Figure. 8 (b), we investigate the influence of hyper-parameter τ in PIC. It shows that the proposed method is sensitive and deteriorates when $\tau \in \{0.01, 0.1\}$. When τ is larger than 1, the model performance tends to stabilize and slightly decline, and the recommended value for τ is 1.0.

5.5.3 Study of AND. In Figue. 9, we evaluate the effectiveness of different hyper-parameters (γ, R) in AND on (a) CIFAR-10 and (b) FLICKR25K datasets, where γ from -1.0 to 1.0 with a step of 0.25 and the maximum R is set to 10, the code length is 64. The diagram in Figure. 9(a) indicates that MAP will increase with R increases and then converges around 3 on CIFAR-10 dataset, and a large γ (≥ 0) might be a good choice. On FLICKR25K, it is noteworthy that when γ is smaller than -0.5, a significant increment would happen in the second round. This is due to the increase of similar pairs, but later proved that this threshold was too loose, leading to model degradation.
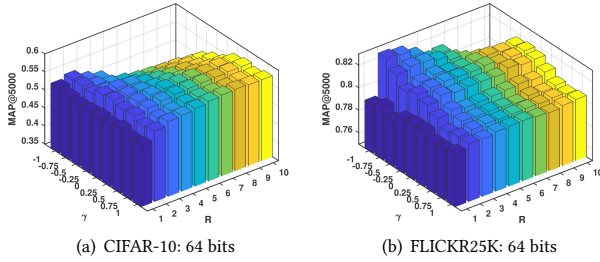

(a) CIFAR-10: 64 bits   (b) FLICKR25K: 64 bits

Figure 9: Parameter Sensitivity of (γ, R) in AND. The code length is set to 64.

5.6 Efficiency Analysis (Q4)

In Table.4, we conduct an experiment to compare the training time of different deep methods on CIFAR-10 dataset. The results indicate that DSAH is efficient and only takes nearly one-third time of the fastest baseline SSDH; Also, DSAH achieves significantly performance with only a fifth of MLS³RDUH's computation time.

Methods	MAP@5000	Training Time (h)
SSDH [49]	0.256	3.0
BGAN [38]	0.587	5.0
MLS³RDUH [43]	0.595	4.8
DSAH	0.622	1.2

Table 4: Training time comparison among the unsupervised deep hashing methods.

5.7 Qualitative Result (Q5)

5.7.1 t-SNE visualization. To better understand the manifold structure of learned hashing code, We compare the t-SNE visualization [29] of BGAN, MLS3RDUH, and DSAH in Figure. 10, in which the data points within the same colors belong to the same class. It could be visually found that DSAH shows a clearer structure, in which we can find some meaningful clusters. Those scatter points within the same class (color) thus own smaller hamming distances with each other.
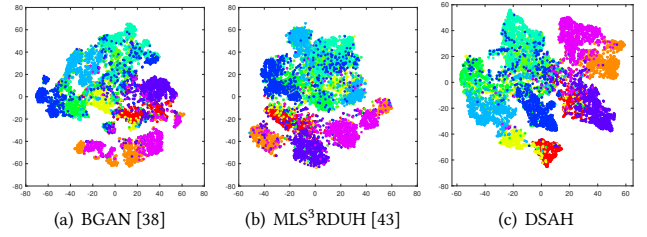

(a) BGAN [38]   (b) MLS³RDUH [43]   (c) DSAH

Figure 10: t-SNE visualization on CIFAR-10 dataset. (64 bits)

5.7.2 Retrieval Result. In Figure 11, we display some retrieval examples that return the top-10 samples based on the Hamming ranking distance on the CIFAR-10 dataset (64-bit). Comparing to the best baseline, our DSAH has fewer fault images. Specially, we found that MLS³RDUH confusing birds in a green scene and flags, while the high-quality prediction of DSAH demonstrates that DSAH could well-distinguish the semantic information of images rather than the low-level feature.
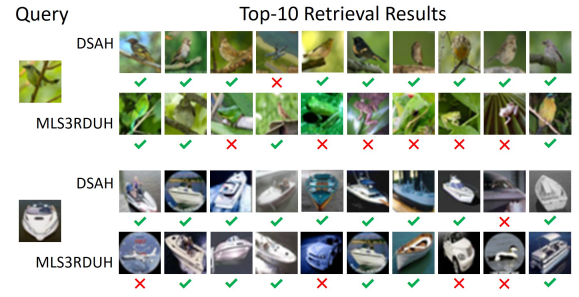

Figure 11: Top-10 retrieved results of DSAH and MLS³RDUH on CIFAR-10 dataset. The green ✓ means that the retrieved samples belong to the same classes of the query images while the red × represents the returned in different classes.

6 CONCLUSIONS

In this paper, we propose a novel deep unsupervised hashing method DSAH to yield binary codes with fully exploring semantic information behind the data pairs. Particularly, DSAH provides two innovative designs: AND and PIC. To overcome the limitation of fixed semantic similarity, we adopt AND technique to refine the pre-computed similarity matrix with the learned representation and adaptively capture the implicit semantic structure behind the data distribution. Further, we employ PIC to distinguish the different importance of data pairs and assign an adjustable weight to each data pair, which fully explores the discriminative information of training data pairs. By combining PIC and AND in one framework, DSAH learns better hash codes in a self-adaptive manner. The extensive experiments on three benchmarks datasets demonstrate the effectiveness of our techniques, and DSAH can achieve competitive performance.

7 ACKNOWLEDGEMENTS

REFERENCES

[1] Alexandr Andoni and Piotr Indyk. 2006. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *2006 47th annual IEEE symposium on foundations of computer science (FOCS'06)*. IEEE, 459–468.

[2] Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. 2016. Deep quantization network for efficient image retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 30.

[3] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. 2017. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*. 5608–5617.

[4] Yudong Chen, Zhihui Lai, Yujuan Ding, Kaiyi Lin, and Wai Keung Wong. 2019. Deep supervised hashing with anchor graph. In *Proceedings of the IEEE International Conference on Computer Vision*. 9796–9804.

[5] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 2009. NUS-WIDE: a real-world web image database from National University of Singapore. In *Proceedings of the ACM international conference on image and video retrieval*. 1–9.

[6] Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. 2017. Stochastic generative hashing. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 913–922.

[7] Cheng Deng, Erkun Yang, Tongliang Liu, Jie Li, Wei Liu, and Dacheng Tao. 2019. Unsupervised semantic-preserving adversarial hashing for image search. *IEEE Transactions on Image Processing* 28, 8 (2019), 4032–4044.

[8] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. 2015. Deep hashing for compact binary codes learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2475–2483.

[9] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *Vldb*, Vol. 99. 518–529.

[10] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2012. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE transactions on pattern analysis and machine intelligence* 35, 12 (2012), 2916–2929.

[11] Jie Gui, Tongliang Liu, Zhenan Sun, Dacheng Tao, and Tieniu Tan. 2017. Fast supervised discrete hashing. *IEEE transactions on pattern analysis and machine intelligence* 40, 2 (2017), 490–496.

[12] Kaiming He, Fang Wen, and Jian Sun. 2013. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2938–2945.

[13] Xiangyu He, Peisong Wang, and Jian Cheng. 2019. K-nearest neighbors hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2839–2848.

[14] Mark J Huiskes and Michael S Lew. 2008. The MIR flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*. 39–43.

[15] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 604–613.

[16] Qing-Yuan Jiang and Wu-Jun Li. 2017. Asymmetric deep supervised hashing. *arXiv preprint arXiv:1707.08325* (2017).

[17] Wang-Cheng Kang, Wu-Jun Li, and Zhi-Hua Zhou. 2016. Column sampling based discrete supervised hashing.. In *AAAI*. 1230–1236.

[18] Weihao Kong and Wu-Jun Li. 2012. Isotropic hashing. In *Advances in neural information processing systems*. 1646–1654.

[19] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[20] Brian Kulis and Kristen Grauman. 2009. Kernelized locality-sensitive hashing for scalable image search. In *2009 IEEE 12th international conference on computer vision*. IEEE, 2130–2137.

[21] Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. 2017. Deep supervised discrete hashing. In *Advances in neural information processing systems*. 2482–2491.

[22] Tong Li, Sheng Gao, and Yajing Xu. 2017. Deep multi-similarity hashing for multi-label image retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 2159–2162.

[23] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. 2015. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855* (2015).

[24] Kevin Lin, Jiwen Lu, Chu-Song Chen, and Jie Zhou. 2016. Learning compact binary descriptors with unsupervised deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1183–1192.

[25] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2011. Hashing with graphs. In *ICML*.

[26] Xuanwu Liu, Zhao Li, Jun Wang, Guoxian Yu, Carlotta Domenicon, and Xiangliang Zhang. 2019. Cross-modal zero-shot hashing. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE, 449–458.

[27] David G Lowe. 1999. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, Vol. 2. Ieee, 1150–1157.

[28] Xin Luo, Liqiang Nie, Xiangnan He, Ye Wu, Zhen-Duo Chen, and Xin-Shun Xu. 2018. Fast scalable supervised hashing. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 735–744.

[29] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[30] Qibing Qin, Lei Huang, Zhiqiang Wei, Kezhen Xie, and Wenfeng Zhang. 2020. Unsupervised Deep Multi-Similarity Hashing with Semantic Structure for Image Retrieval. *IEEE Transactions on Circuits and Systems for Video Technology* (2020).

[31] Zhaofan Qiu, Yingwei Pan, Ting Yao, and Tao Mei. 2017. Deep semantic hashing with generative adversarial networks. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 225–234.

[32] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.

[33] Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal* 27, 3 (1948), 379–423.

[34] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised discrete hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 37–45.

[35] Fumin Shen, Yan Xu, Li Liu, Yang Yang, Zi Huang, and Heng Tao Shen. 2018. Unsupervised deep hashing with similarity-adaptive and discrete optimization. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2018), 3034–3044.

[36] Yuming Shen, Jie Qin, Jiaxin Chen, Mengyang Yu, Li Liu, Fan Zhu, Fumin Shen, and Ling Shao. 2020. Auto-Encoding Twin-Bottleneck Hashing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2818–2827.

[37] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* (2014).

[38] Jingkuan Song, Tao He, Lianli Gao, Xing Xu, Alan Hanjalic, and Heng Tao Shen. 2018. Binary generative adversarial networks for image retrieval. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[39] Shupeng Su, Chao Zhang, Kai Han, and Yonghong Tian. 2018. Greedy hash: Towards fast optimization for accurate hash coding in cnn. In *Advances in neural information processing systems*. 798–807.

[40] Rong-Cheng Tu, Xian-Ling Mao, Bo-Si Feng, and Shu-Ying Yu. 2019. Object detection based deep unsupervised hashing. (2019).

[41] Rong-Cheng Tu, Xian-Ling Mao, Jia-Nan Guo, Wei Wei, and Heyan Huang. 2021. Partial-Softmax Loss based Deep Hashing. In *Proceedings of the Web Conference 2021*. 2869–2878.

[42] Rong-Cheng Tu, Xian-Ling Mao, Bing Ma, Yong Hu, Tan Yan, Wei Wei, and Heyan Huang. 2020. Deep cross-modal hashing with hashing functions and unified hash codes jointly learning. *IEEE Transactions on Knowledge and Data Engineering* (2020).

[43] Rong-Cheng Tu, Xian-Ling Mao, and Wei Wei. 2020. MLS3RDUH: Deep Unsupervised Hashing via Manifold based Local Semantic Similarity Structure Reconstructing. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. 3466–3472.

[44] Bingning Wang, Kang Liu, and Jun Zhao. 2018. Deep Semantic Hashing with Multi-Adversarial Training. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 1453–1462.

[45] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. 2017. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2017), 769–790.

[46] Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. In *Advances in neural information processing systems*. 1753–1760.

[47] Rongkai Xia, Yan Pan, Hanjiang Lai, Cong Liu, and Shuicheng Yan. 2014. Supervised hashing for image retrieval via image representation learning.. In *AAAI*, Vol. 1. 2.

[48] Yan Xia, Kaiming He, Pushmeet Kohli, and Jian Sun. 2015. Sparse Projections for High-Dimensional Binary Codes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[49] Erkun Yang, Cheng Deng, Tongliang Liu, Wei Liu, and Dacheng Tao. 2018. Semantic structure-based unsupervised deep hashing. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 1064–1070.

[50] Erkun Yang, Tongliang Liu, Cheng Deng, Wei Liu, and Dacheng Tao. 2019. Distillhash: Unsupervised deep hashing by distilling data pairs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2946–2955.

[51] Haofeng Zhang, Yifan Gu, Yazhou Yao, Zheng Zhang, Li Liu, Jian Zhang, and Ling Shao. 2020. Deep Unsupervised Self-evolutionary Hashing for Image Retrieval. *IEEE Transactions on Multimedia* (2020).

[52] Wanqian Zhang, Dayan Wu, Yu Zhou, Bo Li, Weiping Wang, and Dan Meng. 2020. Deep Unsupervised Hybrid-similarity Hadamard Hashing. In *Proceedings of the 28th ACM International Conference on Multimedia*. 3274–3282.

[53] Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. 2016. Deep hashing network for efficient similarity retrieval. In *Thirtieth AAAI Conference on Artificial Intelligence*.