Dual Self-Paced Hashing for Image Retrieval

Yuan Sun, Yang Qin, Dezhong Peng, Zhenwen Ren, Chao Yang, and Peng Hu

***Abstract*—In recent years, image hashing has attracted more and more attention in practical retrieval applications due to its low storage cost and high query speed. Although existing hashing methods have achieved promising performance, they always treat both easy and hard points without discrimination, thus easily getting stuck into bad local minima, especially in the presence of noise or outliers. In this paper, we reveal that there exist dual difficulty levels hindering binarization in learning to hash, i.e., samples and bits. To overcome this problem, we propose a novel Dual Self-Paced Hashing method (DSPH) for image retrieval, which learns binary codes by not only evolving from "easy" to "hard" samples but also from "easy" to "hard" bits, mimicking the cognitive learning process from easy to difficult. Specifically, we endow each sample and bit with a weight to estimate the reliability/ease of the row and column, respectively. Then both sample- and bit-level weighting are conducted on rows and columns of Hamming space to enforce the model focus on reliable/easy examples and bits. By gradually increasing the weights during model optimization, more samples and bits are automatically involved in training from "easy" to "hard" via our dual self-paced method, thereby alleviating the adverse impact caused by noises or outliers to learn robust hashing models. Extensive experiments are conducted on four benchmark datasets to demonstrate the superiority and robustness of the proposed DSPH.**

***Index Terms*—Image retrieval, learning to hash, cognitive learning, self-paced learning.**

I. Introduction

RECENTLY, image retrieval [1], [2], [3], [4], [5] has attracted increasing attention from both academia and industry with the explosive growth of images. The core of image retrieval is how to measure the similarities between query and retrieval images. However, traditional continuous-value methods are unsuitable for large-scale image retrieval due to their high storage and computation overhead.

To tackle this problem, numerous hashing learning methods [6], [7], [8], [9] are proposed to binarize the representations, thus embracing lower storage cost and high query speed owing to the binary save and XOR computation. The essential idea of image hashing [6], [10], [11], [12] is to project images into a latent Hamming space while preserving the intrinsic similarities [13], wherein similar images are with lower Hamming distance (i.e., XOR result), and vice versa. More specifically, these methods could be roughly grouped into unsupervised and supervised hashing [14], [15]. Unsupervised hashing learns binary codes by exploiting the intrinsic relationship in the training data, e.g., discrete spectral hashing [16], contrastive hashing [17], and graph hashing [18]. However, their performance is often unsatisfactory due to ignoring semantic information. To mine the discrimination, supervised hashing aims to encapsulate the semantic information into the Hamming space, e.g., matrix factorization hashing [19], supervised asymmetric hashing [20], and similarity preserving hashing [21].

Although the aforementioned methods [22], [23] achieve promising performance, they still face some challenges in learning binary codes. To be specific, most of them implicitly assume that training data are clean while ignoring the existence of noisy samples. However, collecting such clean data is expensive or even impossible in real-world scenarios, thus inevitably leading to noisy data degrading their performance. Furthermore, prior methods always treat both easy and hard samples/bits equally while ignoring the inter-sample and inter-bit differences, which will easily lead to unstable learning and getting stuck into bad local minima, especially in the presence of noise or outliers. Inspired by human cognitive learning, self-paced learning (SPL) [24], [25] was proposed to train the model from 'easy' to 'hard' samples by imitating the human learning process. Thanks to gradual learning, SPL could alleviate the noise/outlier problem and improve generalization, however, it is less touched in learning to hash.

To handle the above problems, we propose a novel **Dual Self-Paced Hashing (DSPH)** for image retrieval in the paper. Our DSPH endows both samples and bits in Hamming space from more reliable/easy to unreliable/hard ones into learning to hash through mimicking human cognitive learning. As shown in Fig. 1, our basic idea is to gradually learn hash codes from 'easy' to 'hard' samples/bits until it is robust enough to handle the difficult ones. Specifically, first, DSPH models the reliability/ease of each sample and each bit by using a specific weight,

Manuscript received 26 April 2023; revised 28 March 2024; accepted 20 April 2024. Date of publication 2 May 2024; date of current version 18 September 2024. This work was supported in part by NSFC under Grant 62372315 and Grant 62102274, in part by Sichuan Science and Technology Program under Grant 2022YFH0021, Grant 24ZDZX0007, Grant 24QYCX0375, Grant 24QYCX0354, Grant 24ZHSF0545, Grant 2023ZYD0143, and Grant 24NSFSC0157, in part by Chengdu Science and Technology Project under Grant 2023-XT00-00004-GX, in part by SCU-LuZhou Sciences and Technology Cooperation Program under Grant 2023CDLZ-16, and in part by Mianyang Science and Technology Program under Grant 2022ZYDF089. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Sanghoon Lee. *(Corresponding authors: Peng Hu; Chao Yang.)*

Yuan Sun, Yang Qin, and Peng Hu are with the College of Computer Science, Sichuan University, Chengdu 610044, China (e-mail: sunyuan_work@163.com; penghu.ml@gmail.com).

Dezhong Peng is with the College of Computer Science, Sichuan University, Chengdu 610044, China, and also with Sichuan Newstrong UHD Video Technology Company, Ltd., Chengdu 610095, China.

Zhenwen Ren and Chao Yang are with the Department of National Defence Science and Technology, Southwest University of Science and Technology, Mianyang 621053, China (e-mail: ychao1983@126.com).

Our source code has been released at https://github.com/sunyuan-cs/DSPH.

Digital Object Identifier 10.1109/TMM.2024.3395969

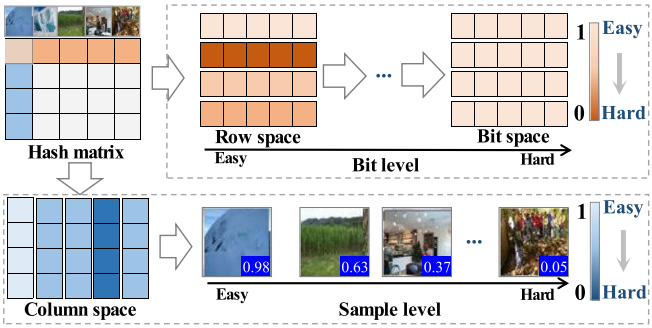


Fig. 1. The basic idea. Inspired by human cognitive learning, we argue learning the binary codes by not only evolving from 'easy' to 'hard' samples but also from 'easy' to 'hard' bits. Hence, the sample- and bit-level self-paced learning could be performed in the row and column space of the hash representation matrix, respectively.

respectively. Second, the sample- and bit-level weights are utilized to make the model focus on reliable/easy samples and bits, respectively. Finally, the dual self-paced regularizer is presented to adaptively optimize the model from 'easy' to 'hard', thereby embracing robust and high-quality codes. The main contributions of this paper are summarized as follows:

- To learn robust and discriminative hash codes, we propose a novel hashing model that alleviate the negative effect of the hard samples and bits in the learning process.
- To the best of our knowledge, this could be the first work that utilizes sample- and bit-level SPL for learning to hash. Different from prior SPL studies, we reveal that such a cognitive learning mechanism is applicable to not only the sample dimension but also the bit dimension in Hamming space.
- Extensive experiments demonstrate the effectiveness and robustness of our DSPH compared to eleven state-of-the-art methods on four widely-used benchmark datasets.

II. Related Work

In this section, we briefly review some related works in supervised hashing and self-paced learning.

A. Supervised Hashing

Recently, numerous supervised hashing methods are proposed to learn discriminative binary codes for retrieval [1], [26], [27]. To fully exploit semantic information, some hashing methods construct a pairwise similarity affinity matrix to capture the relativity between samples. Due to the $n \times n$ pairwise similarity matrix, their space cost is high. For this purpose, fast scalable supervised hashing (FSSH) [28] proposes a pre-computed intermediate term to avoid such a large matrix. Since the balance and decorrelation constraints for unsupervised hashing have been proven to be quite important, strongly constrained discrete hashing (SCDH) [29] imposes these two constraints to improve the quality of hash codes. Most hashing methods ignore the mutual triplet-level ordinal structure, probability ordinal-preserving semantic hashing (POSH) [30] proposes the ordinal-preserving hashing concept based on the non-parametric Bayesian theory. Moreover, they mainly focus on the semantic knowledge and overlook the graph structure and the high-order localities, ordinal-preserving latent graph hashing (OLGH) [31] jointly learns the feature-level locality-similarity preservation and semantic-preserving hash codes.

Inspired by the powerful nonlinear representation of deep neural networks, some deep image hashing methods are proposed. For example, cluster-based weighted distance maximization (CWDM) [32] constructs an adversarial mechanism to stash private images and preserve perceptual similarity. Moreover, to achieve the fine-grained image retrieval task, Deep listwise triplet hashing (DLTH) [33] and sub-region localized hashing (sRLH) [34] are proposed. Inspired by transfer learning, optimal projection guided transfer hashing (GTH) [35] leverages images from the source domain to help learn high-quality hash codes for the target domain. Although the aforementioned methods achieve promising performance, most of them ignore the adverse effect of noise points.

B. Self-Paced Learning

Inspired by the human cognitive learning, self-paced learning (SPL) [36], [37] is proposed to measure the ease level of each sample and weights samples accordingly to enforce model focus on easier samples. With a well-designed self-paced regularizer, the weights are increases from low to high gradually, thus making the model pay more attention to hard samples increasingly, namely from 'easy' to 'hard'. Recently, many SPL-based methods are proposed to explore effective learning orders through different techniques, e.g., smooth multiple kernels [38], ensemble learning [39], etc. For example, prototype-supervised adversarial network (ProS-GAN) [40] utilizes a self-paced weighting learning to replace the previous Hamming distance loss, thereby optimizing the target adversarial samples. Cognitive Multi-modal Consistent Hashing (CMCH) [41] uses self-paced learning to achieve feature aggregation gradually and fuse multi-modal data into a common latent space for multi-modal retrieval.

Although the SPL-based methods achieve promising robustness, almost all of them focus on the contributions of samples to learn representations, while ignoring the diversity between different dimensions. Different from these methods, we explore the self-paced learning in both sample and dimension levels in this paper.

III. Method

In this section, we will elaborate on the proposed DSPH. As illustrated in Fig. 2, DSPH first focuses on easier data, and then gradually focuses on more difficult data until the whole training dataset.

A. Notations

In this paper, the uppercase bold letters and the lowercase bold letters are used to represent the matrices and feature vectors, respectively. $\boldsymbol{I}$ indicates the identity matrix with the corresponding dimensionality. Let $\{\boldsymbol{z}_i, \boldsymbol{y}_i\}_{i=1}^{n}$ be the training image

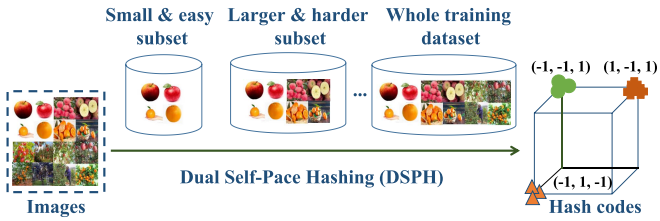


Fig. 2. The basic framework of the proposed DSPH. DSPH mainly models the reliability/ease of each sample and each bit by using a specific weight, respectively. According to the learned reliability/ease, DSPH first trains the hashing model by focusing on 'easy' samples and bits. The weights gradually increase as the learning procedure continues, focusing on more and more difficult samples. Finally, the model encompasses the entire dataset in its training.

data, where $\boldsymbol{z}_i \in \mathbb{R}^{d\times 1}$ is the i-th d-dimensional image sample with the category label $\boldsymbol{y}_i \in \{0,1\}^{c\times 1}$, and c is the number of categories. If $\boldsymbol{z}_i$ belongs to the j-th class, $y_{ji} = 1$ otherwise $y_{ji} = 0$. Besides, let $\boldsymbol{Z} = [\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_n] \in \mathbb{R}^{d\times n}$ and $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n] \in \{0,1\}^{c\times n}$ be image and label matrices, respectively. Image hashing aims to project high-dimensional image features into a discriminative Hamming space, thereby obtaining hash codes $\boldsymbol{B} \in \{-1,1\}^{l\times n}$, where l is the hash length. First, we normalize all samples to be zero mean, i.e., $\sum_{i=1}^{n} \boldsymbol{z}_i = 0$. Then the RBF kernel is conducted on all samples to extract nonlinear local structure information from the data. Specifically, we randomly select k samples as anchor points $\{\boldsymbol{a}_i\}_{i=1}^{k}$ from the training set for memory and computation efficiency. Based on the anchors, the kernel feature of each sample can be computed by

$$\phi(\boldsymbol{z}_i) = \left[exp\left(\frac{\|\boldsymbol{z}_i - \boldsymbol{a}_1\|_2^2}{-2\sigma^2}\right), \ldots, exp\left(\frac{\|\boldsymbol{z}_i - \boldsymbol{a}_k\|_2^2}{-2\sigma^2}\right)\right]^\top, \tag{1}$$

where σ is the kernel width, which could be set as the average Euclidean distances between all samples. To simplify the presentation, we abbreviate $\phi(\boldsymbol{Z})$ to $\boldsymbol{X}$. A common way to learn binary codes $\boldsymbol{B}$ of $\boldsymbol{X}$ is to utilize a linear hash function $\boldsymbol{W}$ as follows:

$$\min_{\boldsymbol{W},\boldsymbol{B}} \|\boldsymbol{W}\boldsymbol{X} - \boldsymbol{B}\|_F^2 \text{ s.t. } \boldsymbol{B} \in \{-1,1\}^{l\times n}. \tag{2}$$

B. Problem Formulation

Noise and outliers are ubiquitous in training data, which will inevitably interfere with data-driven learning methods. However, prior image hashing methods implicitly assume that the training data are clean while ignoring the existence of noisy points. Moreover, they learn hash codes from training data by treating all samples and bits equally, which may be unreasonable due to the inter-sample and inter-bit differences. To tackle this problem, we present a Dual Self-Paced Hashing method (DSPH) for learning to hash from both sample and bit dimensions. Like traditional self-paced learning (SPL) methods, our DSPH automatically incorporates more samples from 'easy' to 'hard' in the training process, which is similar to the way of human cognitive learning, namely, always gradually learning from easy concepts to difficult ones. Different from them, we discover that such a cognitive mechanism is applicable to not only the sample dimension but also the bit dimension in Hamming space. On the one hand, easy samples and bits can be helpful in learning to hash codes; on the other hand, with the learning process, more and more samples and bits become easy for learning. To this end, our DSPH simultaneously performs the sample- and bit-level SPL to gradually train the hashing model from 'easy' to 'hard' until it is powerful enough to handle the complex ones. Accordingly, we could formulate this problem as follows:

$$\begin{aligned}\min_{\boldsymbol{W},\boldsymbol{B},\boldsymbol{s},\boldsymbol{r}} \ & \|\boldsymbol{E}_r(\boldsymbol{W}\boldsymbol{X} - \boldsymbol{B})\boldsymbol{E}_s\|_F^2 + f(\zeta, \boldsymbol{r}, \boldsymbol{s}) \\ s.t.\ & \boldsymbol{B} \in \{-1,1\}^{l\times n}, \quad r_i \in [0,1], s_i \in [0,1], \\ & \boldsymbol{E}_r = diag(\sqrt{r}_1, \sqrt{r}_2, \ldots, \sqrt{r}_l), \\ & \boldsymbol{E}_s = diag(\sqrt{s}_1, \sqrt{s}_2, \ldots, \sqrt{s}_n),\end{aligned} \tag{3}$$

where $f(\zeta, \boldsymbol{r}, \boldsymbol{s})$ is the self-paced regularizer, r_i and s_i are respectively the weights of i-th bit and i-th sample to estimate the reliability along sample and bit dimensions, ζ is a real value that gradually increases with the number of training iterations to dynamically enlarge r_i and s_i. $\boldsymbol{E}_r$ and $\boldsymbol{E}_s$ are two dynamical diagonal matrices to weight samples and bits. In brief, $f(\zeta, \boldsymbol{r}, \boldsymbol{s})$ makes $\boldsymbol{E}_r$ and $\boldsymbol{E}_s$ gradually increase from 'low' to 'high' to enforce sample- and bit-level learning from 'easy' to 'hard' with increasing iterations, respectively. To this end, we follow [42] to formulate $f(\zeta, \boldsymbol{r}, \boldsymbol{s})$ as

$$\begin{aligned} f(\zeta, \boldsymbol{r}, \boldsymbol{s}) &= f(\zeta, \boldsymbol{r}) + f(\zeta, \boldsymbol{s}) \\ &= \sum_{i=1}^{l} \left(1 + e^{-\zeta} - r_i\right) \ln\left(1 + e^{-\zeta} - r_i\right) \\ &\quad + r_i \ln r_i - \zeta r_i + \sum_{i=1}^{n} \left(1 + e^{-\zeta} - s_i\right) \\ &\quad \times \ln\left(1 + e^{-\zeta} - s_i\right) + s_i \ln s_i - \zeta s_i. \end{aligned} \tag{4}$$

As the model is iteratively solved, the loss gradually decreases, making the weights (i.e., r and s) become large. Hence, we think the weight can dynamically assign each sample and bit the easiness/contribution, thus learning from 'easy' to 'hard' with increasing the iterations. More details are given in s-Step and r-Step of the optimization subsection.

Then, we introduce a label projection matrix $\boldsymbol{P}$ to bridge the Hamming space $\boldsymbol{B}$ and label space $\boldsymbol{Y}$ by minimizing the quantization error, thereby encapsulating the discrimination into the binary codes. Thus, we could derive the final objective function as follows:

$$\begin{aligned} \min_{\boldsymbol{W},\boldsymbol{B},\boldsymbol{P},\boldsymbol{s},\boldsymbol{r}} & \|\boldsymbol{E}_r(\boldsymbol{W}\boldsymbol{X} - \boldsymbol{B})\boldsymbol{E}_s\|_F^2 \\ & + \lambda\|\boldsymbol{B} - \boldsymbol{P}\boldsymbol{Y}\|_F^2 + f(\zeta, \boldsymbol{r}, \boldsymbol{s}) \\ s.t.\ & \boldsymbol{P}^\top\boldsymbol{P} = \boldsymbol{I}, \quad \boldsymbol{B} \in \{-1,1\}^{l\times n}, \\ & \boldsymbol{E}_r = diag\left(\sqrt{r}_1, \sqrt{r}_2, \ldots, \sqrt{r}_l\right), r_i \in [0,1], \\ & \boldsymbol{E}_s = diag\left(\sqrt{s}_1, \sqrt{s}_2, \ldots, \sqrt{s}_n\right), s_i \in [0,1], \end{aligned} \tag{5}$$

where λ is a trade-off parameter, and $\boldsymbol{P}$ is the label projection function, which is constrained to be orthogonal to improve discrimination while avoiding trivial solutions.

C. Optimization

Directly optimizing the discrete objective (5) is an NP-hard problem, which is difficult to obtain an optimal solution for all variables. To address the problem, we develop an effective iterative optimization strategy to alternatively solve each variable, i.e., optimizing one variable by fixing all others in each step. The main optimization process is elaborated as follows:

$\boldsymbol{W}$-Step: To solve $\boldsymbol{W}$, by fixing the remaining variables, we could simplify (5) as the following optimization subproblem:

$$\min_{\boldsymbol{W}} \|\boldsymbol{E}_r(\boldsymbol{W}\boldsymbol{X}-\boldsymbol{B})\boldsymbol{E}_s\|_F^2. \tag{6}$$

The above (6) could be rewritten as the trace form of matrix:

$$\min_{\boldsymbol{W}} Tr((\boldsymbol{E}_r(\boldsymbol{W}\boldsymbol{X}-\boldsymbol{B})\boldsymbol{E}_s)(\boldsymbol{E}_r(\boldsymbol{W}\boldsymbol{X}-\boldsymbol{B})\boldsymbol{E}_s)^\top). \tag{7}$$

To solve the above problem, we set the derivative of (7) *w.r.t.* $\boldsymbol{W}$ to 0, and obtain:

$$\boldsymbol{W} = \boldsymbol{B}\boldsymbol{E}_s^2\boldsymbol{X}^\top(\boldsymbol{X}\boldsymbol{E}_s^2\boldsymbol{X}^\top)^{-1}, \tag{8}$$

where $\boldsymbol{E}_s^2 = \boldsymbol{E}_s\boldsymbol{E}_s^\top$.

$\boldsymbol{P}$-Step: We optimize $\boldsymbol{P}$ by fixing the remaining variables in (5), and obtain the following optimization subproblem:

$$\min_{\boldsymbol{P}} \|\boldsymbol{B}-\boldsymbol{P}\boldsymbol{Y}\|_F^2 \quad s.t.\ \boldsymbol{P}^\top\boldsymbol{P} = \boldsymbol{I}. \tag{9}$$

The above (9) is the orthogonal Procrustes problem (OPP) [43], which can be addressed by the singular value decomposition (SVD). Thus, we obtain

$$\boldsymbol{P} = \boldsymbol{U}\boldsymbol{V}^\top, \tag{10}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the left- and right-singular vectors of $\boldsymbol{B}\boldsymbol{Y}^\top$, respectively.

$\boldsymbol{B}$-Step: Similarly, we can obtain the following subproblem of (5) *w.r.t.* $\boldsymbol{B}$:

$$\min_{\boldsymbol{B}} \|\boldsymbol{E}_r(\boldsymbol{W}\boldsymbol{X}-\boldsymbol{B})\boldsymbol{E}_s\|_F^2 + \lambda\|\boldsymbol{B}-\boldsymbol{P}\boldsymbol{Y}\|_F^2, \tag{11}$$

The above (11) could be further rewritten as

$$\max_{\boldsymbol{B}} Tr(\boldsymbol{E}_r\boldsymbol{E}_r^\top\boldsymbol{W}\boldsymbol{X}\boldsymbol{E}_s\boldsymbol{E}_s^\top\boldsymbol{B}^\top + \lambda\boldsymbol{P}\boldsymbol{Y}\boldsymbol{B}^\top). \tag{12}$$

Similar to $\boldsymbol{W}$-Step, the closed-form solution of $\boldsymbol{B}$ can be obtained as follows:

$$\boldsymbol{B} = sgn(\boldsymbol{E}_r^2\boldsymbol{W}\boldsymbol{X}\boldsymbol{E}_s^2 + \lambda\boldsymbol{P}\boldsymbol{Y}). \tag{13}$$

$\boldsymbol{s}$-Step: The subproblem of (5) *w.r.t.* $\boldsymbol{s}$ can be formulated as follows:

$$\min_{\boldsymbol{s}} \|\boldsymbol{E}_r(\boldsymbol{W}\boldsymbol{X}-\boldsymbol{B})\boldsymbol{E}_s\|_F^2 + f(\zeta,\boldsymbol{s})$$
$$s.t.\ \boldsymbol{E}_s = diag(\sqrt{s}_1,\sqrt{s}_2,\ldots,\sqrt{s}_n), s_i\in[0,1]. \tag{14}$$

For the weight of the i-th example, (14) can be simplified as:

$$\min_{\boldsymbol{s}_i\in[0,1]} \boldsymbol{s}_i g_i + f(\zeta,\boldsymbol{s}_i). \tag{15}$$

where $g_i = \|(\boldsymbol{E}_r(\boldsymbol{W}\boldsymbol{X}-\boldsymbol{B}))_{:,i}\|_F^2$ is the quantization loss of the i-th sample. Then let the derivative of (15) *w.r.t.* s_i be zero, one can obtain

$$s_i = \frac{1+e^{-\zeta}}{1+e^{g_i-\zeta}}, \tag{16}$$

Therefore, we can obtain all weights $\{s_i\}_{i=1}^n$ by (16).

$\boldsymbol{r}$-Step: Fixing other variables except $\boldsymbol{r}$, we can rewritten (5) as follows:

$$\min_{\boldsymbol{r}} \|\boldsymbol{E}_r(\boldsymbol{W}\boldsymbol{X}-\boldsymbol{B})\boldsymbol{E}_s\|_F^2 + f(\zeta,\boldsymbol{r})$$
$$s.t.\ \boldsymbol{E}_r = diag(\sqrt{r}_1,\sqrt{r}_2,\ldots,\sqrt{r}_l), r_i\in[0,1]. \tag{17}$$

Similar to $\boldsymbol{s}$, we can obtain

$$r_i = \frac{1+e^{-\zeta}}{1+e^{h_i-\zeta}}, \tag{18}$$

where $h_i = \|((\boldsymbol{W}\boldsymbol{X}-\boldsymbol{V})\boldsymbol{E}_s)_{i,:}\|_F^2$ is the quantization loss of the i-th bit. Then, we can compute all weights $\{r_i\}_{i=1}^l$ by (18).

D. Robustness Analysis

According to [37], if a set of samples or bits has a good fit in the model space, they can be viewed as 'easy'. Therefore, we can use the value of the loss function as a measure criteria, thereby determining the difficulty level of samples or bits. From the (16) and (18), when $g_i<\zeta$ or $h_i<\zeta$, the i-th sample or i-th bit can be implicitly considered as 'easy' with relatively large weights and slow changes, otherwise, regarded as 'hard' with relatively small weights and rapid changes. Moreover, when ζ is fixed, the loss function value only varies rapidly within a certain interval, which directly affects hard samples or bits. To this end, we normalize each loss h_i or g_i by the maximum loss to improve our control over the range of values, thereby aligning the loss of each sample or bit with the rapidly varying interval for ζ. The formula is given as follows:

$$h_i := \frac{h_i}{\max\{h_1,h_2,\ldots,h_l\}},$$
$$g_i := \frac{g_i}{\max\{g_1,g_2,\ldots,g_n\}} \tag{19}$$

To speed up the changes of the weight, we use the step size q to control the increment of ζ at each iteration, i.e., $\zeta = q*\zeta$. Note here that we empirically set $q=1.2$ in the experiments and initialize $\zeta=0.2$. From (16) and (18), one could observe that r_i and s_i are proportional to ζ, i.e., r_i and s_i will increase as ζ increases until approaching one. Hence, in the training process, the weights $\{s_i\}_{i=1}^n$ and $\{r_i\}_{i=1}^l$ are gradually increased, thereby paying more attention to hard samples and bits. Thanks to the property, our model could gradually consider more hard samples from 'easy' to 'hard' to prevent fitting on noise or outliers first, thus embracing stronger robustness.

E. Complexity Analysis

In this section, we study the complexity of the proposed DSPH at each iteration. The computational complexity mainly depends on the five solving processes, i.e., (8), (10), (13),

(16), and (18). Specifically, the computational complexity is $\mathcal{O}(kln)$, $\mathcal{O}(c^2n)$, $\mathcal{O}(kln)$, $\mathcal{O}(kl^2)$, and $\mathcal{O}(kln)$, respectively. Since $k \ll n$, $c \ll n$, and $l \ll n$ for large-scale datasets, the total computational complexity is about $\mathcal{O}(n)$. In addition, since we avoid directly using the $n \times n$ diagonal matrix $\boldsymbol{E}_r$, the space complexity approximates to $\mathcal{O}(n)$. In brief, our DSPH is suitable for large-scale image retrieval.

IV. Experimental Results and Analysis

In this section, we perform various experiments on four widely used image datasets. All comparison methods are conducted on a desktop Windows PC with an Intel Core i9 CPU and 64 GB RAM.

A. Datasets

To evaluate the performance of our DSPH method, we perform experiments on four public datasets. **CIFAR-10** [45] has 10 categories and each with 6,000 images. We extract the 512-dimension GIST features from the original color images with size 32×32. **NUS-WIDE** [50] is collected from the real-world large-scale multi-label image dataset. We select 195,834 images from the 21 most frequent tags. The 500-dimensional bag-of-words features are extracted from original images for our experiments. **Caltech-256** [31] contains 29,780 samples from 256 classes. We extract the 1024-dimensional deep features by using CNN pre-trained on ImageNet, which are obtained from the last fully-connected layer of the neural network. **Place205** [51] is a scene recognition dataset, which has 2.5 M images from 205 classes. In the experiments, 104.1 K images are randomly selected for evaluation. We extract CNN features from the *fc7* layer of an AlexNet pre-trained on ImageNet, and then reduce the features into 128 dimensions by PCA.

In our experiments, on CIFAR-10 and Caltech-256 datasets, we randomly select 1,000 images as the query set, respectively. On NUS-WIDE and Place205 datasets, we randomly select 2,100 and 4,100 images as the query set, respectively. The remaining images are utilized as the training set. For the multi-label dataset, we consider that two images are semantically similar if they share at least one common label, and vice versa.

B. Experimental Setting

To evaluate the effectiveness of our DSPH, we compare it with eleven state-of-the-art hashing methods, including SDH [44], FSDH [45], FSSH [28], SSLH [46], RSLH [47], SCDH [29], POPSH [30], OLGH [31], SASH [27], LBSE [48], and SCLCH [49]. For all baselines, we adopt source codes provided by the original authors and use the suggested parameters from their published papers. For our method, we set experimentally the hyperparameter $\lambda = 100$ on all datasets. Empirically, we set $k = 1500$, $q = 1.2$, and $\zeta = 0.2$. Besides, to evaluate the robustness of the proposed model, we also randomly add Gaussian white noise with 0.01 variance and zero mean to image features with different noise rates (i.e., 10%, 20%, 30%, and 40%). In addition, to provide a more nuanced understanding of DSPH's resilience, we extend robust experiments to other types of noise noise on the first two datasets. Specifically, we add the salt-and-pepper noise with a noise density of 0.05 to image features with different noise rates (i.e., 10%, 20%, 30%, and 40%). SSLH, RSLH, and SASH have the $n \times n$ similarity matrix, which can result in the high computation and space complexity. Hence, we randomly choose 10500 images on the NUS-WIDE dataset as training set and sample 5000 images on other datasets as one.

C. Evaluation Protocols

To comprehensively evaluate retrieval performance, four popular evaluation protocols are utilized, i.e., mAP, Precision@TopN, NDCG@Top100, and precision-recall curve. Notably, mAP is the mean of the average precision scores for all query samples. Precision@TopN represents Precision scores for the top N samples. NDCG@100 represents normalized discounted cumulative gain at rank 100. The precision-recall curve can be used for evaluating the precision, recall, and the number of retrieved points. For CIFAR-10 and NUS-WIDE datasets, they contain more than 5000 images in each category. Hence, following [30], [31], we use Precision@Top5000 as the retrieval evaluation metric. For Caltech-256 and Place205, they contain less than 5000 images in each category, only a few hundred. Therefore, for uniform presentation, we use Precision@Top50 to show the retrieval results.

D. Comparison Shallow Results

We report mAP, Precision@TopN, and NDCG@Top100 with different bits on four benchmark datasets in Tables I–III, respectively. From these results, one could draw the following observations:

1) When using different evaluation protocols, our proposed DSPH can obtain the best retrieval performance on all datasets compared with competitors. For example, when the bit length is 128 bits, the mAP scores improved by 1.41%, 2.71%, 2.25%, and 0.93%, respectively. These experimental results demonstrate the effectiveness of the dual self-paced hashing framework.
2) As the hash length increases, there is a trend towards better retrieval performance for all comparison methods on four large-scale datasets. It indicates that long-length hash codes can provide more discriminative information to support the retrieval task.
3) For all datasets except the NUS-WIDE dataset, Precision@TopN score is lower than the mAP score. It indicates that the system is performing well in the top results but performing poorly beyond the top N results. The difference between mAP and Precision@5000 could indicate the retrieval system's strengths and weaknesses in retrieving relevant results, particularly in the top or bottom portions of the search results.
4) We can find that when the number of samples is large (for example NUS-WIDE), the data becomes more complex,

TABLE I
THE MAP RESULTS (%) OF DIFFERENT BIT LENGTH ON FOUR DATASETS

Method	CIFAR-10				NUS-WIDE				Caltech-256				Place205			
	16	32	64	128	16	32	64	128	16	32	64	128	16	32	64	128
SDH [44]	38.56	42.92	45.43	47.26	31.20	31.45	35.08	37.51	28.91	39.71	47.82	53.82	12.90	18.59	23.45	26.28
FSDH [45]	38.02	42.75	45.47	47.20	31.44	31.44	31.44	31.45	29.44	39.78	47.09	53.44	13.14	18.72	23.45	26.57
FSSH [28]	61.78	65.38	68.25	69.06	55.37	60.86	61.08	64.40	41.14	52.73	62.50	68.41	29.65	40.47	46.70	51.50
SSLH [46]	33.60	37.86	39.80	37.79	41.93	42.23	43.32	43.84	9.30	9.73	12.90	38.98	1.51	2.52	4.21	15.13
RSLH [47]	28.81	34.06	34.86	36.22	41.23	42.49	41.74	43.35	20.89	30.25	38.32	45.03	6.89	11.28	15.42	19.26
SCDH [29]	67.75	70.43	71.63	72.14	46.64	47.04	47.75	47.71	43.54	55.65	63.55	67.91	27.74	27.84	28.04	28.06
POPSH [30]	64.33	67.94	69.14	70.15	57.46	60.80	61.14	62.38	51.86	62.25	69.80	74.48	36.55	46.64	51.35	53.34
OLGH [31]	66.40	68.31	64.22	49.83	53.64	56.80	55.67	52.38	47.77	60.12	62.00	56.82	34.41	43.57	37.41	27.06
SASH [27]	21.84	21.85	26.65	25.61	34.93	35.93	39.28	/	20.89	27.81	33.79	36.47	6.17	10.17	13.05	11.24
LBSE [48]	68.21	69.52	72.02	72.14	61.62	63.71	63.93	64.59	51.90	64.02	71.47	75.63	36.05	45.78	51.47	53.39
SCLCH [49]	68.13	70.27	70.68	71.95	61.83	62.59	63.41	63.70	50.23	63.36	70.00	73.88	35.99	45.48	51.32	53.78
Our DSPH	**69.29**	**72.63**	**73.19**	**73.55**	**62.17**	**64.99**	**66.20**	**67.11**	**53.11**	**64.89**	**71.99**	**76.73**	**37.18**	**46.68**	**52.37**	**54.27**

/ denotes that the running time are too high. The highest scores are in bold.

TABLE II
THE PRECISION@TOPN RESULTS (%) OF DIFFERENT BIT LENGTH ON FOUR DATASETS

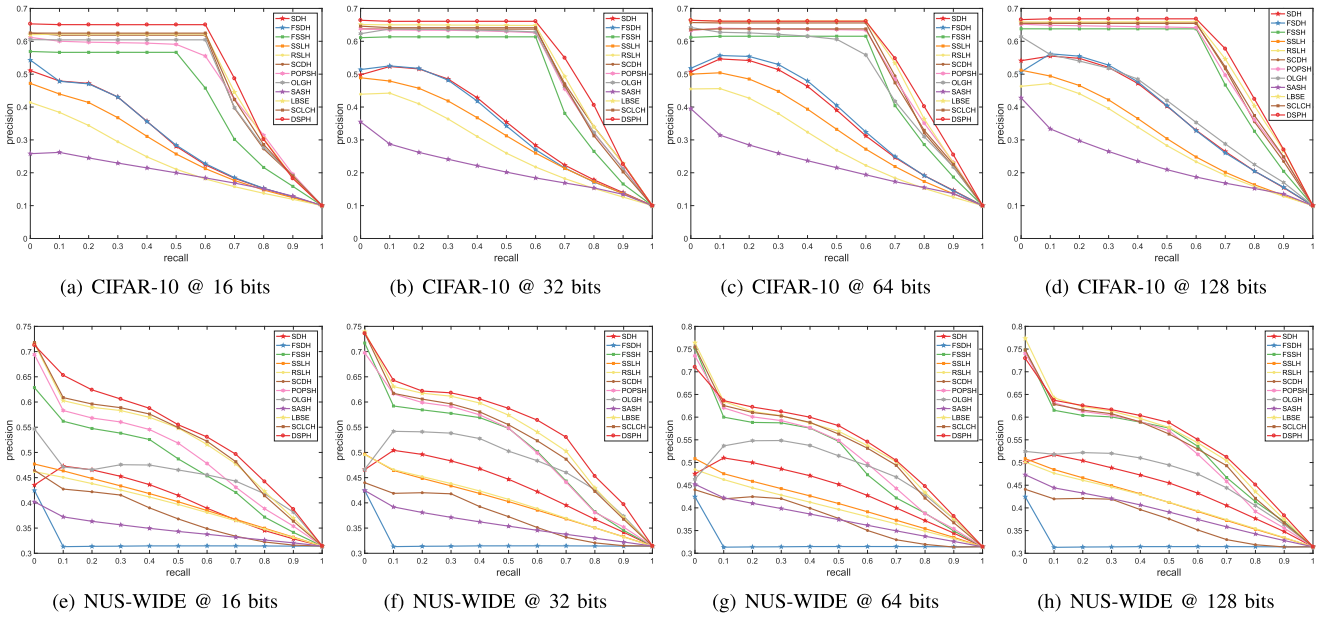
Method	CIFAR-10 (N=5000)				NUS-WIDE (N=5000)				Caltech-256 (N=50)				Place205 (N=50)			
	16	32	64	128	16	32	64	128	16	32	64	128	16	32	64	128
SDH [44]	46.26	50.13	52.00	53.98	30.89	31.38	32.16	35.47	40.35	53.30	61.19	66.32	24.73	32.65	37.13	39.05
FSDH [45]	45.76	50.39	52.03	53.90	31.65	31.66	31.65	31.69	40.89	53.45	60.59	65.57	25.13	32.90	37.01	39.22
FSSH [28]	55.13	58.76	61.46	62.14	61.61	63.16	64.29	66.34	39.11	50.59	59.44	65.23	27.80	37.49	42.67	46.27
SSLH [46]	44.49	47.97	49.48	50.76	47.76	48.10	49.19	50.36	10.56	11.33	15.69	53.68	2.07	4.44	9.39	31.11
RSLH [47]	39.88	44.93	46.47	47.58	46.42	48.47	49.51	50.67	30.69	43.44	53.53	59.76	15.77	25.55	31.46	35.48
SCDH [29]	61.86	63.70	65.58	65.53	42.14	41.34	41.12	40.93	41.59	53.44	60.96	64.09	26.82	27.02	27.05	27.02
POPSH [30]	59.71	63.29	63.77	64.63	61.81	64.55	65.12	66.00	41.59	53.44	60.96	64.09	33.66	42.52	45.86	46.99
OLGH [31]	60.44	63.56	62.11	64.03	47.54	52.47	50.16	52.53	45.80	58.58	62.87	62.90	31.89	40.64	39.51	36.42
SASH [27]	28.49	30.30	39.23	39.09	39.65	41.90	45.67	/	27.62	36.29	44.59	53.53	10.50	18.06	24.27	30.22
LBSE [48]	62.53	62.67	65.49	65.22	64.15	67.27	68.37	68.59	49.96	62.11	68.99	72.52	33.78	42.74	46.99	48.23
SCLCH [49]	62.42	64.41	63.87	65.04	64.70	64.95	66.56	66.64	48.31	61.35	67.11	70.57	33.54	42.55	47.05	47.81
Our DSPH	**63.97**	**66.81**	**66.55**	**66.93**	**66.97**	**68.45**	**69.18**	**70.12**	**51.24**	**62.88**	**69.41**	**73.45**	**34.29**	**43.08**	**48.17**	**49.71**

/ denotes that the running time are too high. The highest scores are in bold.

TABLE III
THE NDCG@TOP100 RESULTS (%) OF DIFFERENT BIT LENGTH ON FOUR DATASETS

Method	CIFAR-10				NUS-WIDE				Caltech-256				Place205			
	16	32	64	128	16	32	64	128	16	32	64	128	16	32	64	128
SDH [44]	47.93	51.99	53.30	54.90	28.32	28.74	29.16	30.26	34.77	45.98	53.98	59.11	24.08	32.05	36.67	38.50
FSDH [45]	47.65	52.21	53.45	54.74	27.75	27.73	27.73	27.73	35.45	46.27	53.04	58.49	24.44	32.27	36.59	38.76
FSSH [28]	54.99	58.71	61.44	62.15	46.49	52.38	52.94	55.28	39.24	50.80	59.63	65.38	27.46	37.50	42.67	46.29
SSLH [46]	44.31	47.97	49.76	50.86	36.51	37.14	38.29	39.34	10.59	11.15	15.03	45.94	2.11	4.34	8.97	29.42
RSLH [47]	39.85	44.79	46.63	47.90	35.22	36.86	37.87	38.81	26.34	36.76	45.42	51.78	14.93	23.84	29.85	33.82
SCDH [29]	61.92	63.69	65.58	65.51	33.97	33.48	33.49	33.19	41.69	53.55	61.13	64.28	26.82	27.02	27.06	27.02
POPSH [30]	60.61	63.29	63.86	65.02	45.81	48.02	50.19	52.58	50.06	60.04	66.89	71.20	33.68	42.52	45.86	46.96
OLGH [31]	60.50	63.80	63.16	59.10	38.60	35.18	36.72	41.26	45.99	58.21	61.22	58.83	31.87	40.65	39.03	35.01
SASH [27]	27.97	39.39	38.23	37.92	30.16	32.07	34.57	/	25.16	32.37	39.33	44.72	10.10	16.92	22.35	27.18
LBSE [48]	62.51	62.70	65.46	65.22	46.14	48.08	48.55	48.77	50.28	62.09	69.20	72.75	33.74	42.73	46.97	48.22
SCLCH [49]	62.43	64.44	63.88	65.05	46.02	47.82	47.25	49.92	48.52	61.54	67.35	70.81	33.55	42.52	47.05	47.82
Our DSPH	**63.95**	**66.76**	**66.60**	**66.91**	**52.29**	**53.58**	**55.24**	**54.87**	**51.50**	**63.04**	**69.55**	**73.69**	**34.90**	**43.48**	**48.28**	**48.39**

/ denotes that the running time are too high. The highest scores are in bold.



(a) CIFAR-10 @ 16 bits (b) CIFAR-10 @ 32 bits (c) CIFAR-10 @ 64 bits (d) CIFAR-10 @ 128 bits

(e) NUS-WIDE @ 16 bits (f) NUS-WIDE @ 32 bits (g) NUS-WIDE @ 64 bits (h) NUS-WIDE @ 128 bits

Fig. 3. The Precision-Recall curves for different bits on four datasets.

which causes that image retrieval becomes more challenging. When the number of classes is larger, the evaluation scores on Caltech-256 are much better than those on Place205, since the number of samples on Caltech-256 is lower than that on Place205.

5) On the Place205 dataset (large classes and large-scale data), SSLH, RSLH, and SCDH are poor, which indicates that they cannot resist interference with large classes and large-scale data.

In addition, we illustrate the precision-recall curves with different bits in Fig. 3. From these figures, the following conclusions can be reached:

1) Compared with these baselines on all datasets, our DSPH shows superior capabilities to handle large-scale image retrieval tasks and obtain the best performance with different bit lengths.
2) With the hash length increases, the precision scores also increase under each recall value. In addition, the advantage of performance is more obvious when hash codes are long bits. It indicates long hash codes can provide more semantic information.
3) We can find that precision-recall curves of our DSPH decline more slowly. It indicates our DSPH has strong stability in retrieving similar samples. The precision results of all hashing methods have a large drop with each recall on the Place205, compared to NUS-WIDE. This may be because the images on Place205 have the great intra-class variations.

E. Robustness Experiments

To further show the robustness of our proposed DSPH against noise, we perform robustness analysis experiments with 64 bits. The mAP, Precision@TopN, and NDCG@Top100 scores with Gaussian noise on four benchmark datasets are shown in Tables IV–VI. And we have the following observations:

1) DSPH not only achieves the best retrieval performance without noise, but also keeps superior robustness under noise. For example, for different noise with 64-bit lengths, DSPH improves the mAP scores by 3.08%, 4.67%, 4.40%, and 4.09% on NUS-WIDE, respectively. And DSPH improves the mAP scores by 2.82%, 2.88%, 3.13%, and 4.84% on CIFAR-10, respectively. When the bit length is short, the performance of comparison methods is poor on hard datasets.
2) We find that some methods (e.g., SSLH and POPSH) fail to handle Caltech-256 under noise. This may be due to the fact that the noise remarkably increases the difficulty of hard data (e.g., with the large class number), making them hard to learn high-quality hash codes.
3) On all datasets, as the noise increases, the performance of the comparison methods decreases sharply, which indicates these methods have weak robustness. The performance of our DSPH falls slowly, which indicates some resistance to noise.
4) DSPH remarkably outperforms all baselines under all noise rates, which demonstrates its strong robustness against noise. That is, our dual self-paced method could effectively alleviate the negative impact caused by noises and outliers to learn the robust hashing model.

F. Ablation Analysis

In this section, we conduct ablation studies to investigate the contributions of the different components to our DSPH on CIFAR-10 under 0% and 40% noise, respectively. Specifically, we design two variants of our DSPH: 1) sample-level self-paced hashing (SSPH) with only $\boldsymbol{E}_r$, and 2) bit-level self-paced hashing (BSPH) with only $\boldsymbol{E}_s$. The experimental results are shown

TABLE IV
THE MAP RESULTS (%) OF 64 BITS ON FOUR DATASETS WITH DIFFERENT PERCENTAGES OF GAUSSIAN NOISE

Method	CIFAR-10				NUS-WIDE				Caltech-256				Place205			
	10%	20%	30%	40%	10%	20%	30%	40%	10%	20%	30%	40%	10%	20%	30%	40%
SDH [44]	36.05	29.42	24.7	21.05	30.47	30.43	31.76	31.42	32.63	24.7	20.10	16.12	22.44	19.90	19.90	18.15
FSDH [45]	36.23	29.82	24.78	21.02	31.00	31.09	30.8	31.48	32.33	25.03	20.52	15.83	22.71	21.28	20.08	18.96
FSSH [28]	63.11	59.61	56.46	54.29	60.25	58.43	57.72	56.14	47.49	41.53	35.57	29.22	45.70	45.15	44.07	42.36
SSLH [46]	27.87	23.34	19.93	16.79	36.31	36.52	37.52	36.43	9.36	8.01	3.94	4.81	2.25	9.40	8.33	6.04
RSLH [47]	28.13	22.95	19.41	16.77	36.48	35.78	34.80	34.97	24.53	17.12	13.20	10.30	14.89	13.50	11.93	10.13
SCDH [29]	63.39	59.76	55.95	52.87	47.10	47.15	47.70	47.99	43.13	25.14	18.17	12.54	27.87	25.55	26.47	23.61
POPSH [30]	61.78	52.32	47.67	42.33	36.84	35.91	37.86	35.21	45.35	11.86	1.96	1.33	50.25	50.14	48.86	47.31
OLGH [31]	57.10	52.66	48.96	46.09	32.47	32.90	32.97	32.82	45.21	29.44	22.59	13.76	36.54	35.37	34.01	29.13
SASH [27]	19.66	17.13	15.49	14.74	31.51	31.36	31.86	31.99	20.74	14.21	11.54	7.45	13.35	12.38	11.68	9.28
LBSE [48]	67.31	63.00	59.60	54.07	39.71	40.17	40.87	41.58	61.18	52.45	45.80	38.36	50.41	50.06	48.78	47.49
SCLCH [49]	66.69	62.49	58.18	55.07	44.41	44.90	46.62	47.16	60.09	50.36	44.04	**39.77**	50.64	49.63	48.49	47.79
Our DSPH	**69.58**	**64.62**	**61.79**	**58.46**	**63.33**	**63.10**	**62.12**	**60.23**	**61.48**	**53.21**	**45.84**	38.56	**51.24**	**50.27**	**49.28**	**49.18**

The highest scores are in bold.

TABLE V
THE PRECISION@TOPN RESULTS (%) OF 64 BITS ON FOUR DATASETS WITH DIFFERENT PERCENTAGES OF GAUSSIAN NOISE

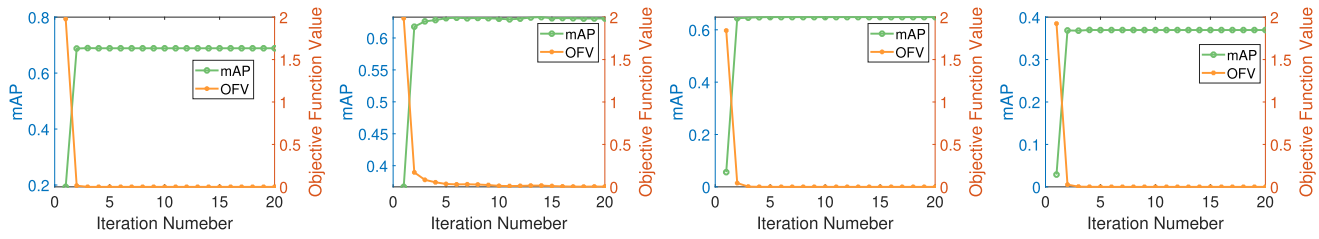
Method	CIFAR-10				NUS-WIDE				Caltech-256				Place205			
	10%	20%	30%	40%	10%	20%	30%	40%	10%	20%	30%	40%	10%	20%	30%	40%
SDH [44]	43.87	36.87	31.47	26.79	31.24	29.56	26.53	27.95	46.15	36.96	31.79	25.87	36.6	34.76	34.76	33.74
FSDH [45]	44.08	37.34	31.48	26.54	29.58	28.85	25.73	28.90	45.81	37.51	32.26	25.19	37.22	36.18	35.18	33.87
FSSH [28]	55.39	51.49	47.97	45.97	60.62	55.73	54.47	52.24	44.14	38.35	32.46	25.81	41.74	40.93	39.81	37.76
SSLH [46]	42.40	35.98	31.28	23.96	35.39	37.16	39.10	36.43	13.69	12.09	7.55	9.69	6.20	23.68	22.81	18.29
RSLH [47]	42.89	36.43	31.44	25.42	37.14	36.91	35.43	36.55	37.44	27.61	21.83	17.86	31.08	30.51	28.81	26.32
SCDH [29]	55.59	51.74	47.39	43.92	38.35	38.28	39.51	40.31	39.76	22.52	16.14	11.20	26.88	24.56	25.53	22.58
POPSH [30]	55.21	45.01	39.57	33.68	37.88	36.43	35.97	27.84	41.42	9.24	0.31	0.20	45.75	44.85	43.42	41.79
OLGH [31]	55.05	50.61	46.33	43.11	19.65	18.74	18.89	18.72	48.35	32.72	25.78	16.08	39.39	38.73	37.42	33.10
SASH [27]	28.25	23.20	19.89	19.01	26.22	25.82	29.32	28.72	30.48	22.94	19.55	13.66	24.75	24.14	24.15	21.77
LBSE [48]	60.06	54.63	50.87	45.43	24.08	35.26	27.48	26.62	58.31	50.38	41.86	34.81	45.86	45.46	44.26	42.80
SCLCH [49]	59.16	54.21	48.95	45.71	32.27	33.36	37.04	33.34	57.33	46.98	40.62	36.39	46.34	45.20	43.90	43.05
Our DSPH	**62.88**	**57.51**	**54.00**	**50.27**	**63.55**	**62.31**	**59.97**	**56.68**	**62.39**	**54.17**	**46.62**	**39.10**	**47.05**	**45.82**	**44.79**	**44.64**

The highest scores are in bold.

TABLE VI
THE NDCG@TOP100 RESULTS (%) OF 64 BITS ON FOUR DATASETS WITH DIFFERENT PERCENTAGES OF GAUSSIAN NOISE

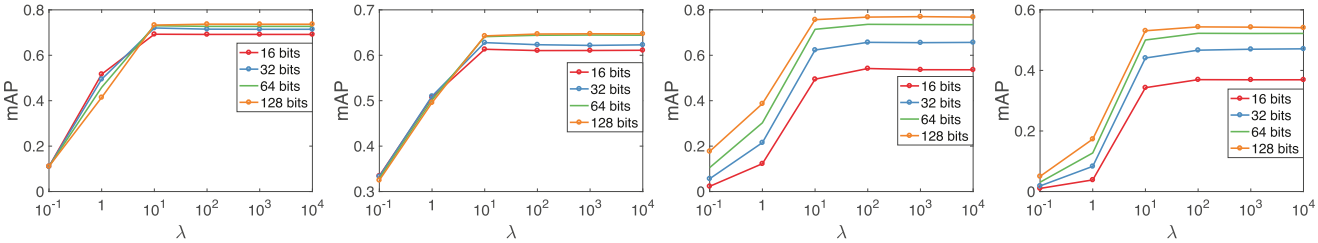
Method	CIFAR-10				NUS-WIDE				Caltech-256				Place205			
	10%	20%	30%	40%	10%	20%	30%	40%	10%	20%	30%	40%	10%	20%	30%	40%
SDH [44]	47.56	42.4	37.68	32.96	26.75	25.46	21.43	26.35	39.48	31.44	26.74	22.16	35.87	33.48	33.48	31.98
FSDH [45]	47.74	42.84	37.51	32.52	24.54	24.21	21.29	25.08	39.09	31.79	27.21	21.74	36.3	35.03	33.75	32.23
FSSH [28]	55.34	51.43	47.93	45.99	50.84	46.01	45.75	44.56	44.38	38.6	32.58	26.09	41.73	40.90	39.79	37.78
SSLH [46]	42.09	35.87	30.64	23.58	25.68	26.66	28.83	26.90	12.58	11.46	6.75	8.74	5.48	21.40	20.20	16.21
RSLH [47]	43.01	36.43	31.18	24.94	29.13	28.63	27.68	28.25	31.32	23.15	18.64	15.32	29.00	28.27	26.24	23.62
SCDH [29]	55.52	51.69	47.32	43.86	31.70	31.77	32.88	33.72	39.87	22.62	16.21	11.17	26.88	24.56	25.53	22.58
POPSH [30]	56.01	44.58	39.11	33.14	29.88	28.01	27.61	20.50	41.67	9.30	0.35	0.20	45.73	44.84	43.39	41.80
OLGH [31]	57.58	52.88	48.70	44.83	17.65	16.72	16.34	16.81	45.66	30.70	23.90	14.78	38.84	38.68	36.71	32.30
SASH [27]	27.43	22.71	19.41	18.73	21.43	20.93	23.59	23.18	26.42	19.75	17.10	11.90	22.82	22.08	21.90	19.34
LBSE [48]	60.00	54.77	51.21	45.45	24.19	25.84	27.07	25.64	58.45	50.63	42.13	35.11	45.84	45.46	44.26	42.81
SCLCH [49]	59.19	54.25	49.02	45.80	24.99	25.77	28.02	25.19	57.52	47.22	40.82	36.70	46.33	45.18	43.89	43.04
Our DSPH	**62.88**	**57.42**	**53.92**	**50.12**	**51.98**	**51.17**	**49.39**	**46.65**	**60.78**	**52.35**	**45.19**	**37.92**	**47.04**	**45.83**	**44.79**	**44.62**

The highest scores are in bold.



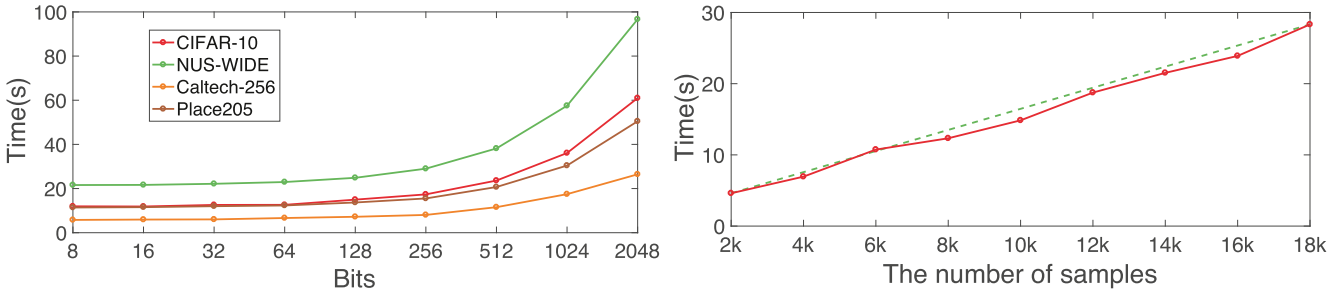
(a) CIFAR-10 (b) NUS-WIDE (c) Caltech-256 (d) Place205

Fig. 4. Convergence curves and mAP with 16 bits on all datasets.



(a) CIFAR-10 (b) NUS-WIDE (c) Caltech-256 (d) Place205

Fig. 5. Parameter analysis results for different bits on different datasets.



(a) Time w.r.t. bits (b) Time w.r.t. samples

Fig. 6. Training time comparison.

TABLE VII
ABLATION ANALYSIS (MAP: %) ON CIFAR-10

Noise	0%				40%			
Method	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
SSPH	69.00	72.14	72.87	72.56	49.87	52.28	55.58	57.22
BSPH	68.45	72.32	72.43	72.79	49.18	52.12	56.33	57.03
DSPH	**69.29**	**72.63**	**73.19**	**73.55**	**52.88**	**56.04**	**58.46**	**58.76**

The highest scores are in bold.

in Table VII, wherein one could observe that all components contribute to improving the retrieval performance.

G. Convergence Experiments

To further investigate the convergence of our DSPH, we draw the convergence curves with 16 bits on all datasets as shown in Fig. 4. Note here that OFV is Objective Function Value. From the figure, one could see that the objective function values stably become smaller as the number of iterations increases on each dataset, and achieve convergence rapidly within 5 iterations. Moreover, the mAP scores also rapidly increase and become stabilized.

H. Parameter Sensitivity Analysis

Unlike prior arts with many hyper-parameters, as shown in Table VIII, our DSPH involves only one to tune, i.e., λ. For example, previous methods usually need to tune more parameters, which will remarkably increase the overhead of manual tuning. Particularly, the comparison methods invariably require two hyper-parameters, and some methods even reach five. Therefore, fewer parameters are more applicable for real-world large-scale image retrieval tasks. To investigate the sensitivity of our method, we draw the performance curves vs. different λ in Fig. 5. From the figure, one could observe that the mAP scores are gradually improved until stable as λ increases. DSPH performs well when λ is in the large range of $[10^1, \ldots, 10^4]$, thus indicating that our DSPH is insensitive to λ in the range.

TABLE VIII
THE NUMBER OF HYPER-PARAMETERS AND THE TRAINING TIME (SECOND) WITH 64 BITS HASH CODES

Methods	Num	CIFAR-10	NUS-WIDE	Caltech-256	Place205
SDH [44]	2	16.30	101.52	20.69	52.64
FSDH [45]	2	3.38	16.46	2.37	7.62
FSSH [28]	3	2.86	9.35	2.37	5.64
SSLH [46]	5	50.38	58.58	60.70	59.58
RSLH [47]	5	6.54	27.56	6.51	6.38
SCDH [29]	3	38.87	98.07	18.22	20.04
POPSH [30]	2	0.58	4.04	0.30	0.93
OLGH [31]	3	2.35	6.54	1.51	3.56
SASH [27]	4	1893.75	11701.05	1787.95	1872.21
LBSE [48]	4	10.83	35.58	9.75	19.97
SCLCH [49]	2	1.45	4.29	1.14	3.29
Our DSPH	1	12.62	22.95	6.65	12.34

I. Training Time

In Fig. 6, we draw the time curves about bit length and the number of training samples. As shown in Fig. 6(a), the training time remains stable when the hash bit length is less than 256. However, when the hash length is more than 256 bits, training time shows a great increase. In terms of bit length, computational complexity is square relative to bit length (i.e., $\mathcal{O}(l^2)$). Moreover, Fig. 6(b) shows that our DSPH has linear time complexity related to the number of training samples. It further verifies that the computational complexity of our DSPH is about $\mathcal{O}(n)$. The running time of our DSPH depends mainly on training time and test time. The test time of all comparison methods depends on the process of calculating Hamming distance, so they all have the same test time. Thereupon, we just have to focus on training time. On four datasets, the training time (second) with 64 bits hash codes is shown in Table VIII. From the results, the training time of our DSPH is comparable to that of LBSE. Obviously, we propose dual self-paced learning to learn high-quality hash codes without sacrificing too much training time. Moreover, on the large-scale NUS-WIDE, the training time is moderate compared with the comparison methods. It indicates that our DSPH can be applied to large-scale retrieval tasks.

V. CONCLUSION

In this paper, we propose a Dual Self-Paced Hashing method (DSPH) to alleviate the negative impact caused by noises and outliers. Specifically, DSPH adopts both sample- and bit-level weighting in Hamming space to estimate the reliability of samples and bits. Then, the sample- and bit-level SPLs are simultaneously performed to gradually train the hashing model from 'easy' to 'hard' until it is powerful enough to handle the hard ones, thus remitting to the noise and boosting the retrieval performance. Comprehensive experiments on four benchmark datasets demonstrate the effectiveness and robustness of the proposed DSPH. The potential limitations of our DSPH is that the retrieval performance degrades due to a large amount of information loss when hash bits are extremely short. In the future, we plan to extend our DSPH to stream data application scene, and further study online self-paced hashing.

REFERENCES

[1] L. Zhu, Z. Huang, Z. Li, L. Xie, and H. T. Shen, "Exploring auxiliary context: Discrete semantic transfer hashing for scalable image retrieval," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5264–5276, Nov. 2018.

[2] Z. Li, J. Tang, L. Zhang, and J. Yang, "Weakly-supervised semantic guided hashing for social image retrieval," *Int. J. Comput. Vis.*, vol. 128, pp. 2265–2278, 2020.

[3] Z. Li, J. Tang, and T. Mei, "Deep collaborative embedding for social image understanding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 9, pp. 2070–2083, Sep. 2019.

[4] W. Tang et al., "Context disentangling and prototype inheriting for robust visual grounding," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 5, pp. 3213–3229, May 2024.

[5] Z. Li and J. Tang, "Weakly supervised deep metric learning for community-contributed image retrieval," *IEEE Trans. Multimedia*, vol. 17, pp. 1989–1999, 2015.

[6] Y. Cao et al., "Learning to hash with dimension analysis based quantizer for image retrieval," *IEEE Trans. Multimedia*, vol. 23, pp. 3907–3918, 2021.

[7] P. Hu, X. Wang, L. Zhen, and D. Peng, "Separated variational hashing networks for cross-modal retrieval," in *Proc. 27th ACM Int. Conf. Multimedia*, 2019, pp. 1721–1729.

[8] H. Cui, L. Zhu, J. Li, Y. Yang, and L. Nie, "Scalable deep hashing for large-scale social image retrieval," *IEEE Trans. Image Process.*, vol. 29, pp. 1271–1284, 2020.

[9] P. Hu et al., "Joint versus independent multiview hashing for cross-view retrieval," *IEEE Trans. Cybern.*, vol. 51, no. 10, pp. 4982–4993, Oct. 2021.

[10] C. Deng et al., "Unsupervised semantic-preserving adversarial hashing for image search," *IEEE Trans. Image Process.*, vol. 28, no. 8, pp. 4032–4044, Aug. 2019.

[11] L. Yuan et al., "Central similarity quantization for efficient image and video retrieval," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 3083–3092.

[12] Y. Sun et al., "Relaxed energy preserving hashing for image retrieval," *IEEE Trans. Intell. Transp. Syst.*, early access, Jan. 25, 2024, doi: 10.1109/TITS.2024.3351841.

[13] W. Kang, W. Li, and Z. Zhou, "Column sampling based discrete supervised hashing," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 1230–1236.

[14] Y. Sun, D. Peng, and Z. Ren, "Discrete aggregation hashing for image set classification," *Expert Syst. With Appl.*, vol. 237, 2024, Art. no. 121615.

[15] Y. Sun, X. Wang, D. Peng, Z. Ren, and X. Shen, "Hierarchical hashing learning for image set classification," *IEEE Trans. Image Process.*, vol. 32, pp. 1732–1744, 2023.

[16] D. Hu, F. Nie, and X. Li, "Discrete spectral hashing for efficient similarity retrieval," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1080–1091, Mar. 2019.

[17] P. Hu et al., "Unsupervised contrastive cross-modal hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 3, pp. 3877–3889, Mar. 2023.

[18] W. Liu, C. Mu, S. Kumar, and S. Chang, "Discrete graph hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3419–3427.

[19] S. He et al., "Bidirectional discrete matrix factorization hashing for image search," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 4157–4168, Sep. 2020.

[20] Z. Zhang et al., "Scalable supervised asymmetric hashing with semantic and latent factor embedding," *IEEE Trans. Image Process.*, vol. 28, no. 10, pp. 4803–4818, Oct. 2019.

[21] M. Lin, R. Ji, S. Chen, X. Sun, and C.-W. Lin, "Similarity-preserving linkage hashing for online image retrieval," *IEEE Trans. Image Process.*, vol. 29, pp. 5289–5300, 2020.

[22] Y. Sun, D. Peng, J. Dai, and Z. Ren, "Stepwise refinement short hashing for image retrieval," in *Proc. 31st ACM Int. Conf. Multimedia*, 2023, pp. 6501–6509.

[23] Y. Sun, Z. Ren, P. Hu, D. Peng, and X. Wang, "Hierarchical consensus hashing for cross-modal retrieval," *IEEE Trans. Multimedia*, vol. 26, pp. 824–836, 2024.

[24] J. Li, Z. Kang, C. Peng, and W. Chen, "Self-paced two-dimensional PCA," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 8392–8400.

[25] X. Wang, Y. Chen, and W. Zhu, "A survey on curriculum learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 4555–4576, Sep. 2022.

[26] X. Nie, X. Liu, J. Guo, L. Wang, and Y. Yin, "Supervised discrete multiple-length hashing for image retrieval," *IEEE Trans. Big Data*, vol. 9, no. 1, pp. 312–327, Feb. 2023.

[27] Y. Shi, X. Nie, X. Liu, L. Zou, and Y. Yin, "Supervised adaptive similarity matrix hashing," *IEEE Trans. Image Process.*, vol. 31, pp. 2755–2766, 2022.
[28] X. Luo et al., "Fast scalable supervised hashing," in *Proc. 41st Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2018, pp. 735–744.
[29] Y. Chen, Z. Tian, H. Zhang, J. Wang, and D. Zhang, "Strongly constrained discrete hashing," *IEEE Trans. Image Process.*, vol. 29, pp. 3596–3611, 2020.
[30] Z. Zhang, X. Zhu, G. Lu, and Y. Zhang, "Probability ordinal-preserving semantic hashing for large-scale image retrieval," *ACM Trans. Knowl. Discov. From Data*, vol. 15, no. 3, pp. 1–22, 2021.
[31] Z. Zhang and C.-M. Pun, "Learning ordinal constraint binary codes for fast similarity search," *Inf. Process. Manage.*, vol. 59, no. 3, 2022, Art. no. 102919.
[32] Y. Xiao, C. Wang, and X. Gao, "Evade deep image retrieval by stashing private images in the hash space," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9651–9660.
[33] Y. Liang, Y. Pan, H. Lai, W. Liu, and J. Yin, "Deep listwise triplet hashing for fine-grained image retrieval," *IEEE Trans. Image Process.*, vol. 31, pp. 949–961, 2022.
[34] X. Xiang, Y. Zhang, L. Jin, Z. Li, and J. Tang, "Sub-region localized hashing for fine-grained image retrieval," *IEEE Trans. Image Process.*, vol. 31, pp. 314–326, 2022.
[35] L. Zhang et al., "Optimal projection guided transfer hashing for image retrieval," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 30, no. 10, pp. 3788–3802, Oct. 2020.
[36] X. Wu, J. Chang, Y.-K. Lai, J. Yang, and Q. Tian, "BiSPL: Bidirectional self-paced learning for recognition from web data," *IEEE Trans. Image Process.*, vol. 30, pp. 6512–6527, 2021.
[37] F. Ma, D. Meng, X. Dong, and Y. Yang, "Self-paced multi-view co-training," *J. Mach. Learn. Res.*, vol. 21, no. 57, pp. 1–38, 2020.
[38] Q. Zhang, Z. Kang, Z. Xu, S. Huang, and H. Fu, "Spaks: Self-paced multiple kernel subspace clustering with feature smoothing regularization," *Knowl.-Based Syst.*, vol. 253, 2022, Art. no. 109500.
[39] P. Zhou et al., "Self-paced clustering ensemble," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 4, pp. 1497–1511, Apr. 2021.
[40] Z. Zhang, X. Wang, G. Lu, F. Shen, and L. Zhu, "Targeted attack of deep hashing via prototype-supervised adversarial networks," *IEEE Trans. Multimedia*, vol. 24, pp. 3392–3404, 2022.
[41] J. An, H. Luo, Z. Zhang, L. Zhu, and G. Lu, "Cognitive multi-modal consistent hashing with flexible semantic transformation," *Inf. Process. Manage.*, vol. 59, no. 1, 2022, Art. no. 102743.
[42] C. Xu, D. Tao, and C. Xu, "Multi-view self-paced learning for clustering," in *Proc. 24th Int. Conf. Artif. Intell.*, 2015, pp. 3974–3980.
[43] J. C. Gower and G. B. Dijksterhuis, *Procrustes Problems*, vol. 30. Oxford, U.K.: OUP Oxford, 2004.
[44] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 37–45.
[45] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Fast supervised discrete hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 490–496, Feb. 2018.
[46] X. Liu et al., "Supervised short-length hashing," in *Proc. Int. Joint Conf. Artif. Intell.*, 2019, pp. 3031–3037.
[47] X. Liu et al., "Reinforced short-length hashing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 31, no. 9, pp. 3655–3668, Sep. 2021.
[48] X. Liu et al., "Learning binary semantic embedding for large-scale breast histology image analysis," *IEEE J. Biomed. Health Inform.*, vol. 26, no. 7, pp. 3240–3250, Jul. 2022.
[49] J. Qin et al., "Joint specifics and consistency hash learning for large-scale cross-modal retrieval," *IEEE Trans. Image Process.*, vol. 31, pp. 5343–5358, 2022.
[50] P. Hu, Z. Huang, D. Peng, X. Wang, and X. Peng, "Cross-modal retrieval with partially mismatched pairs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 8, pp. 9595–9610, Aug. 2023.
[51] F. Cakir, K. He, S. A. Bargal, and S. Sclaroff, "Hashing with mutual information," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2424–2437, Oct. 2019.