Similarity-Preserving Linkage Hashing for Online Image Retrieval

Mingbao Lin, Rongrong Ji, *Senior Member, IEEE*, Shen Chen, Xiaoshuai Sun, and Chia-Wen Lin, *Fellow, IEEE*

***Abstract*— Online image hashing aims to update hash functions on-the-fly along with newly arriving data streams, which has found broad applications in computer vision and beyond. To this end, most existing methods update hash functions simply using discrete labels or pairwise similarity to explore intra-class relationships, which, however, often deteriorates search performance when facing a domain gap or semantic shift. One reason is that they ignore the particular semantic relationships among different classes, which should be taken into account in updating hash functions. Besides, the common characteristics between the label vectors (can be regarded as a sort of binary codes) and to-be-learned binary hash codes have left unexploited. In this paper, we present a novel online hashing method, termed Similarity Preserving Linkage Hashing (SPLH), which not only utilizes pairwise similarity to learn the intra-class relationships, but also fully exploits a latent linkage space to capture the inter-class relationships and the common characteristics between label vectors and to-be-learned hash codes. Specifically, SPLH first maps the independent discrete label vectors and binary hash codes into a linkage space, through which the relative semantic distance between data points can be assessed precisely. As a result, the pairwise similarities within the newly arriving data stream are exploited to learn the latent semantic space to benefit binary code learning. To learn the model parameters effectively, we further propose an alternating optimization algorithm. Extensive experiments conducted on three widely-used datasets demonstrate the superior performance of SPLH over several state-of-the-art online hashing methods.**

***Index Terms*— Image retrieval, online hashing, binary codes, similarity preservation.**

I. Introduction

RAPID and unstoppable exploration of visual data brings significant challenges to traditional image hashing methods [1]–[20] that resort to learning fixed hash functions without any model updating and adaptation. Recently, online hashing has attracted ever-increasing research focus due to its high flexibility, which leverages newly arriving data to update hash functions efficiently, while preserving essential information carried in existing data streams. In principle, online hashing merits in two-fold: (1) It preserves the advantages of traditional hashing methods: the low storage cost and efficiency of pairwise distance computation in the Hamming space. (2) It can be trained efficiently for large-scale applications, as its hash functions can be updated instantly based on newly arriving streaming data.

Essentially, online hashing distills the core information of the newly arriving data stream while preserving the information of the past ones. The works of online hashing can be classified into supervised methods and unsupervised methods. Among the supervised methods, representative works include, just to name a few, Online Kernel Hashing (OKH) [21], Adaptive Hashing (AdaptHash) [22], Online Supervised Hashing (OSH) [23], Online Hashing with Mutual Information (MIHash) [24], Hadamard Codebook based Online Hashing (HCOH) [25], and Balanced Similarity for Online Discrete Hashing (BSODH) [26]. For instance, OKH [21], AdaptHash [22], MIHash [24], and BSODH [26] all utilize label information to define the similarity (dissimilarity) between two data samples. By contrast, HCOH [25] and OSH [23], assign a pre-defined ECOC codebook learned from discrete labels. Representative unsupervised online hashing methods include SketchHash [27] and FROSH [28], *etc*. These unsupervised approaches consider the inherent statistical properties among data, *e.g.*, distribution and variance, which are efficient but lack flexibility and adaptivity. Instead, supervised online hashing typically yields better performance by leveraging label information, which is studied in this paper.

Despite the exciting progress, the accuracy of existing online hashing methods is still far from satisfactory. In particular, we identify two major issues that need to be further tackled: (1) As shown in Fig. 1 (left side), existing supervised methods simply resort to discrete labels or (intra-class) pairwise similarity in learning hash functions, while neglecting the (inter-class) semantic relationships that are particularly important when facing problems like a domain gap [29] and semantic shift [30]. (2) A label vector can be by nature regarded as a hashing code for each data sample. Hence, there exists a correlation between label vectors and binary codes, which is worth further exploration in learning hash functions.

For the first issue, semantic relationships were explored in several non-online hashing methods [30], [31], which, however, cannot be directly applied to online hashing since input data arrive in a streaming manner, thereby making

Manuscript received October 11, 2019; revised February 19, 2020; accepted March 13, 2020. Date of publication March 24, 2020; date of current version March 31, 2020. This work was supported in part by the Nature Science Foundation of China under Grant U1705262, Grant 61772443, Grant 61572410, Grant 61802324, and Grant 61702136, in part by the National Key R&D Program under Grant 2017YFC0113000 and Grant 2016YFB1001503, and in part by the Nature Science Foundation of Fujian Province, China, under Grant 2017J01125 and Grant 2018J01106. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Jiwen Lu. *(Corresponding author: Rongrong Ji.)*

Mingbao Lin, Rongrong Ji, Shen Chen, and Xiaoshuai Sun are with the Media Analytics and Computing Laboratory, Department of Artificial Intelligence, School of Informatics, Xiamen University, Xiamen 361005, China (e-mail: rrji@xmu.edu.cn).

Chia-Wen Lin is with the Department of Electrical Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan, and also with the Institute of Communications Engineering, National Tsing Hua University, Hsinchu 30013, Taiwan.

Digital Object Identifier 10.1109/TIP.2020.2981879

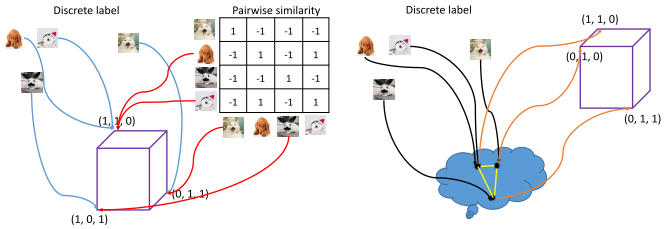


Fig. 1. Illustration of the difference between the proposed SPLH and existing methods [21]–[26]. The existing methods (left side) simply consider discrete labels or (intra-class) pairwise similarity in learning hash functions, while neglecting the (inter-class) semantic relationships. For example, a dog is more semantically related to a cat than that to a rabbit. By contrast, SPLH maps the discrete label vectors into a latent linkage space to capture the inter-class semantic relationships (right side), through which the semantic-related images are closer in the Hamming space. (Best view with zooming in).

the inter-class relationship hard to capture. Besides, existing methods neglect the similarity between different classes to capture the inter-class relationships. For the second issue, capturing the characteristics between label vectors and binary codes remains an open problem. To the best of our knowledge, the label coding methods [17], [32] map label vectors to binary codes by LSH [33], and then use the transformed label vectors as target codes to guide the learning of hash functions. However, the straightforward usage of label vectors has not been exploited.

To address the above problems, we argue that both the *inter- & intra-class relationships* and the *common characteristics* shared between label vectors and learned binary codes are vital. In this paper, we propose a novel supervised online hashing method, termed **S**imilarity **P**reserving **L**inkage **H**ashing (SPLH), to consider the above designs for supervised online hashing. SPLH innovates in three-fold: First, as shown in Fig. 1 (right side), to capture the inter-class relationships and the common characteristics between label vectors and binary codes, we propose to map the discrete label vectors and binary codes to a linkage space where the semantic distances between data points can be assessed precisely. Second, unlike the existing approaches that exploit the similarity between the newly arriving streaming data and the set of past data [26], SPLH uses a more practical way that exploits the relationships among the new streaming data, since the fast-growing size of accumulated past data makes it inefficient to exploit the similarity based on the historic dataset. Lastly, we propose an efficient alternating optimization algorithm to solve the non-convex online learning of binary codes.

The rest of this paper is organized as follows: Sec. II surveys important related works. In Sec. III, we elaborate the framework of the proposed method. Sec. IV analyzes the model complexity. The experimental results and analyses are provided in Sec. V. And finally, we conclude this paper in Sec. VI.

II. Related Works

In this section, we briefly review two related topics: offline hashing and online hashing.

A. Offline Hashing

Offline hashing updates hash functions in a batch manner where a collection of training data is given in advance. To this end, data-independent Locality Sensitivity Hashing (LSH) [33] and its variants [34]–[36] resort to preserving the cosine similarities between samples by using random Gaussian projections to map samples into binary features. Usually, the similarities can be preserved for sufficiently long codes. An alternative is the data-dependent methods. For example, Spectral Hashing (SH) [1] learns hash functions based on spectral graph partitioning: first learning the eigenvectors from the Laplacian of a training dataset and then binarizing low-dimensional real values. Anchor Graph Hashing (AGH) [37] performs clustering on a training dataset to reduce the storage space and computational complexity for constructing the graph Laplacian. Besides, the similarity is computed via k-nearest neighbor anchors which can preserve the local structure in a low-dimensional subspace to a certain extent. ITerative Quantization (ITQ) [4] learns an orthogonal rotation matrix to refine the initial projection matrix learned by PCA so as to minimize the quantization error of mapping the data to the vertices of binary hypercube. Scalable Graph Hashing (SGH) [38] approximates a graph through feature transformation without explicitly computing the similarity graph matrix, based on which a sequential learning method is proposed to learn the hash functions in a bit-wise manner. Ordinal Constraint Hashing (OCH) [39] learns rank-preserving features with a graph-based approximation to embed the ordinal relations. It reduces the size of ordinal graph while keeping the ordinal relations through a small data set via clusters or random samples. Liu *et al.* [40] proposed a self-improvement hash learning framework where the different code sets are obtained for the training data, and then two fusion strategies are adopted to fuse these codes into a single set.

Most of the above hashing methods are originally designed and experimented on hand-crafted features, their poor representation power usually limits their practical applications. Recently, more works have been focusing on utilizing the power of deep convolutional neural networks (CNNs) [41]–[43] to boost hashing performance. Xia *et al.* [44] proposed a two-step supervised hashing method that learns deep CNN based hash functions with pre-computed binary codes. To characterize the non-linear relationship among samples, Lu *et al.* [45] proposed a deep neural network to exploit multiple hierarchical non-linear transformations to learn compact binary codes. Deep Cauchy Hashing (DCH) [46] devised a pairwise cross-entropy loss based on the Cauchy distribution for efficient Hamming space retrieval. Yang *et al.* [47] distilled the relationship between the initial noisy similarity signals and the semantic similarity labels, based on which a Bayesian learning framework was built to learn hash functions. To support fine-grained retrieval applications, Deep Saliency Hashing (DSaH) [48] introduces a saliency mechanism into hashing, which adopts an attention network to mine discriminative regions.

B. Online Hashing

By contrast, online hashing updates hash functions in a streaming manner by which the training data come sequentially. Supervised online hashing makes use of label

information to learn discriminative hash functions. Online Kernel Hashing [21] is among the first of this kind, that requires sample pairs to update hash functions via an online passive-aggressive strategy [49]. Adaptive Hashing [22] defines a hinge-like loss, which is approximated by a differentiable Sigmoid function to update hash functions with stochastic gradient descent (SGD). In [23], a general two-step hashing scheme was introduced, which first assigns binary Error Correcting Output Codes (ECOC) [50]–[52] to labeled data, and then learns a set of hash functions to fit the binary ECOC using Online Boosting. Fatih *et al*. [24] proposed an Online Hashing with Mutual Information, which aims at optimizing the mutual information between the neighbors and non-neighbors given a query. Lin *et al*. proposed a Hadamard Codebook based Online Hashing (HCOH), where a more discriminative Hadamard matrix [53] is used as the ECOC codebook to guide the learning of hash functions through linear regression [25] or classification [54]. Balanced Similarity for Online Discrete Hashing (BSODH) [26] was recently proposed to investigate the correlation between a new data stream and the existing dataset via an inner-product fashion. To address the data imbalance problem, BSODH adopts two equilibrium factors to balance the similarity matrix and enables the application of discrete optimization [9] to online learning. Different from the above works that focus on efficiently updating hash functions, Weng [55] proposed updating the binary codes of a dataset by projecting them into another binary space while fixing the hash functions.

Existing unsupervised online hashing schemes were mainly based on the idea of "data sketch" [56], where a small size of sketch data is leveraged to preserve the main property of a large-scale dataset. To this end, Leng *et al*. [27] proposed an Online Sketching Hashing (OSH), that employs an efficient variant of Singular Value Decomposition (SVD) to learn hash functions, with a PCA-based batch learning on the sketch to learn hashing weights. A computationally efficient version of OSH was proposed in [28], where the independent Subsampled Randomized Hadamard Transform (SRHT) is applied on different data chunks to make the sketch more compact yet accurate, so as to accelerate the sketching process.

Nevertheless, unsupervised online hashing suffers low performance due to the lack of label supervisions. Though great progress has been made, [21]–[26], supervised online hashing cannot well capture the label information. To explain, the semantic relationships among different classes and the common characteristics between label vectors and binary codes have not been well explored yet, as discussed in Sec. I.

III. Proposed Method

A. Problem Definition

We denote the set of training samples as $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$, where each column is a d-dimensional feature vector and n is the number of training samples. The goal of hashing is to learn a set of binary hash codes $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_n] \in \mathbb{R}^{k\times n}$, where $\mathbf{b}_i \in \{-1, +1\}^k$ and $k(\ll d)$ is the code length. Under the supervised setting, a class label set $\mathbf{L} = [\mathbf{l}_1, \ldots, \mathbf{l}_n] \in \{-1, +1\}^{c\times n}$ is also given, where $\mathbf{l}_i \in \{-1, +1\}^c$ is the label vector of $\mathbf{x}_i$ and c is the total number of categories. If $\mathbf{x}_i$ belongs to a category, the corresponding element in vector $\mathbf{l}_i$ is labeled with 1, and -1 otherwise. A common way to obtain the binary values of $\mathbf{X}$ is to utilize a linear function followed by a sign function as follows:

$$\mathbf{F}(\mathbf{X}) = \mathrm{sgn}(\mathbf{W}^T\mathbf{X}), \tag{1}$$

where $\mathbf{W} = [\mathbf{w}_1, \ldots, \mathbf{w}_k] \in \mathbb{R}^{d\times k}$ is the projection matrix and $\mathrm{sgn}(u)$ returns 1 if $u > 0$, and -1 otherwise.

In the online setting, dataset $\mathbf{X}$ cannot be available once for all. Instead, at round t, the hash functions are updated based on a new data chunk $\mathbf{X}^t = [\mathbf{x}_1^t, \ldots, \mathbf{x}_{n_t}^t] \in \mathbb{R}^{d\times n_t}$ added to the dataset. Correspondingly, let $\mathbf{B}^t = [\mathbf{b}_1^t, \ldots, \mathbf{b}_n^t] \in \{-1, +1\}^{k\times n_t}$ denote the learned binary codes for $\mathbf{X}^t$, $\mathbf{L}^t = [\mathbf{l}_1^t, \ldots, \mathbf{l}_{n_t}^t] \in \{-1, +1\}^{c\times n_t}$ is the class label set, and $\mathbf{W}^t$ is the updated projection matrix, where n_t is the size of data at round t. Similarly, the hash functions updated at round t can be rewritten as follows:

$$\mathbf{F}^t(\mathbf{X}^t) = \mathrm{sgn}\big((\mathbf{W}^t)^T\mathbf{X}^t\big), \tag{2}$$

We summarize the notations used in this paper in Table I, which will be detailed in the following contexts.

TABLE I
Notations and Their Meanings

Notation	Meaning
$\mathbf{X}$	$\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathbb{R}^{d\times n}$: Set of n training samples; each column of $\mathbf{X}$ corresponds to one sample
$\mathbf{X}^t$	$\mathbf{X} = [\mathbf{x}_1^t, ..., \mathbf{x}_{n_t}^t] \in \mathbb{R}^{d\times n_t}$: Set of n_t training samples at the t-th round; each column of $\mathbf{X}^t$ corresponds to one sample
$\mathbf{B}$	$\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_n] \in \mathbb{R}^{k\times n}$: Binary code matrix for $\mathbf{X}$
$\mathbf{B}^t$	$\mathbf{B}^t = [\mathbf{b}_1^t, ..., \mathbf{b}_{n_t}^t] \in \mathbb{R}^{k\times n_t}$: Binary code matrix for $\mathbf{X}^t$
$\mathbf{L}$	$\mathbf{L} = [\mathbf{l}_1, ..., \mathbf{l}_n] \in \{0, +1\}^{c\times n}$: Label set for $\mathbf{X}$
$\mathbf{L}^t$	$\mathbf{L}^t = [\mathbf{l}_1^t, ..., \mathbf{l}_n^t] \in \{0, +1\}^{c\times n_t}$: Label set for $\mathbf{X}^t$
$\mathbf{W}$	$\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_k] \in \mathbb{R}^{d\times k}$: Projection matrix for the learned hashing functions
$\mathbf{W}^t$	$\mathbf{W}^t = [\mathbf{w}_1^t, ..., \mathbf{w}_k^t] \in \mathbb{R}^{d\times k}$: Projection matrix updated at the t-th round
$\mathbf{C}^t$	$\mathbf{C}^t \in \mathbb{R}^{c\times r}$: projection matrix for $\mathbf{L}^t$
$\mathbf{U}^t$	$\mathbf{U}^t \in \mathbb{R}^{k\times r}$: Projection matrix for $\mathbf{B}^t$
$\mathbf{V}^t$	$\mathbf{V}^t \in \mathbb{R}^{r\times n_t}$: Latent semantic space
k	Number of hashing bits
c	Total number of categories
d	Dimension of features
r	Dimension of latent semantic space
$\|\cdot\|_F$	Frobenius norm

B. Problem Formulation

The framework of SPLH is shown in Fig. 2. Generally, this framework contains two steps: building a latent linkage space in Fig. 2(c) and constructing similarity-preserving hash codes in Fig. 2(e). Specifically, the label vectors and binary codes are mapped into a latent linkage space to explore the semantic relationships among different classes, and the common characteristics between the label vectors and binary codes. Furthermore, constructing similarity-preserving hash codes aims to exploit the relationships among the newly arriving data streams.

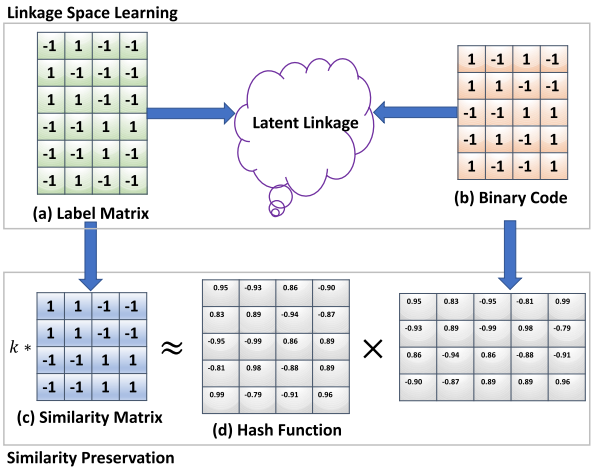


Fig. 2. Framework of the proposed SPLH. The top row shows the linkage space learning. The label vectors (a) and to-be-learned binary codes (b) are mapped into a latent linkage space where the semantic relationships among different classes (inter-class relationships) and the common characteristics between the label vectors and binary codes can be well preserved. The bottom shows the similarity preservation. The similarity matrix (c) is constructed according to the label matrix. The learned binary codes are used as a target to guide the learning of hashing functions (d). Similarity preservation is exploited among the newly arriving streaming data (intra-class relationship) by preserving the inner product between hash function and the similarity matrix (k is the code length).

1) Linkage Space Learning: Suppose there exists a to-be-learned linkage space $\mathbf{V}^t \in \mathbb{R}^{r\times n_t}$ [30], [31] that preserves the semantic relationships among different classes, and the common characteristics shared between discrete label vectors and to-be-learned binary codes, where r denotes the dimension of the latent semantic space. To learn the linkage space, we first map the discrete labels and binary codes into this space, through which the semantic distances among data points can be measured more accurately:

$$\begin{aligned}&\mathcal{L}_1^t(\mathbf{B}^t,\mathbf{C}^t,\mathbf{U}^t,\mathbf{V}^t)\\&=\left\|(\mathbf{C}^t)^T\mathbf{L}^t-(\mathbf{U}^t)^T\mathbf{B}^t\right\|_F^2\\&\quad+\left\|(\mathbf{C}^t)^T\mathbf{L}^t-\mathbf{V}^t\right\|_F^2+\left\|(\mathbf{U}^t)^T\mathbf{B}^t-\mathbf{V}^t\right\|_F^2\\&\quad+\sigma\left(\left\|\mathbf{C}^t\right\|_F^2+\left\|\mathbf{U}^t\right\|_F^2\right)\quad s.t.\ \mathbf{B}^t\in\{-1,+1\}^{k\times n_t},\end{aligned}\tag{3}$$

where σ is a regularization parameter. $\mathbf{C}^t \in \mathbb{R}^{c\times r}$ and $\mathbf{U}^t \in \mathbb{R}^{k\times r}$ are two projection matrices, that map the discrete label set $\mathbf{L}^t$ and binary codes $\mathbf{B}^t$, respectively, into the latent linkage space. As a result, the relative semantic distances can be preserved in the latent linkage space $\mathbf{V}^t$. Besides, the learning of $\mathbf{B}^t$ can well capture the semantic relationships among different classes, leading to more discriminative hashing codes.

To further encode the out-of-sample data points, we propose learning the hash functions by minimizing the error between the hash functions $\mathbf{F}^t(\mathbf{X}^t)$ and the learned binary codes $\mathbf{B}^t$ through the Frobenius norm. As a result, Eq. (3) can be rewritten as

$$\begin{aligned}&\mathcal{L}_1^t(\mathbf{B}^t,\mathbf{C}^t,\mathbf{U}^t,\mathbf{V}^t,\mathbf{W}^t)\\&=\left\|(\mathbf{C}^t)^T\mathbf{L}^t-(\mathbf{U}^t)^T\mathbf{B}^t\right\|_F^2\\&\quad+\left\|(\mathbf{C}^t)^T\mathbf{L}^t-\mathbf{V}^t\right\|_F^2+\left\|(\mathbf{U}^t)^T\mathbf{B}^t-\mathbf{V}^t\right\|_F^2\\&\quad+\left\|\mathbf{F}^t(\mathbf{X}^t)-\mathbf{B}^t\right\|_F^2+\sigma\left(\left\|\mathbf{C}^t\right\|_F^2+\left\|\mathbf{U}^t\right\|_F^2\right)\\&\quad s.t.\ \mathbf{B}^t\in\{-1,+1\}^{k\times n_t}.\end{aligned}\tag{4}$$

2) Similarity Preservation: Although Eq. (4) can learn the inter-class semantic relationships and the common characteristics between label vectors and binary codes, the pairwise similarity between samples is not exploited. Essentially, if two samples are similar in the original feature space, they should share similar binary codes in the to-be-learned Hamming space, and vice versa. This can be achieved by the prevailing scheme of minimizing the inner product between hashing codes and their corresponding similarity matrix [5], [9], [57], which has proven to be effective in online learning [26]. Nevertheless, [26] considers the asymmetric correlation between a newly arriving data chunk and the past streaming data, which, however, becomes computationally unaffordable when the size of past streaming data grows rapidly.

Instead, we consider the symmetric relationship between sample pairs in a newly arriving data chunk $\mathbf{X}^t$. To this end, at round t, we construct two data batches, $\mathbf{S}_A^t \in \mathbf{X}^t$ and $\mathbf{S}_B^t \in \mathbf{X}^t$, as similar pairs (*i.e.*, the i-th columns of $\mathbf{S}_A^t$ and $\mathbf{S}_B^t$ are similar to each other), and $\mathbf{D}_A^t \in \mathbf{X}^t$ and $\mathbf{D}_B^t \in \mathbf{X}^t$ as dissimilar pairs (*i.e.*, the i-th columns of $\mathbf{D}_A^t$ and $\mathbf{D}_B^t$ are dissimilar). The objective function for pairwise similarity preservation is then defined as

$$\begin{aligned}\mathcal{L}_2^t(\mathbf{W}^t)=&\sum_i\left\|\left(\mathbf{F}^t\left((\mathbf{S}_A^t)_{.i}\right)\right)^T\mathbf{F}^t\left((\mathbf{S}_B^t)_{.i}\right)-k\right\|_F^2\\&+\sum_i\left\|\left(\mathbf{F}^t\left((\mathbf{D}_A^t)_{.i}\right)\right)^T\mathbf{F}^t\left((\mathbf{D}_B^t)_{.i}\right)+k\right\|_F^2,\end{aligned}\tag{5}$$

where $(\mathbf{S}_A^t)_{.i}$, $(\mathbf{S}_B^t)_{.i}$, $(\mathbf{D}_A^t)_{.i}$, $(\mathbf{D}_B^t)_{.i}$ denote the i-th columns of $\mathbf{S}_A^t$, $\mathbf{S}_B^t$, $\mathbf{D}_A^t$ and $\mathbf{D}_B^t$, respectively.

However, the existence of $\mathrm{sgn}(u)$ function in Eq. (2) makes Eq. (4) and Eq. (5) NP-hard. To address the complexity problem, we approximate the discrete $\mathrm{sgn}(u)$ function using the continuous $\tanh(u)$, and modify Eq. (4) as follows:

$$\begin{aligned}&\hat{\mathcal{L}}_1^t(\mathbf{B}^t,\mathbf{C}^t,\mathbf{U}^t,\mathbf{V}^t,\mathbf{W}^t)\\&=\left\|(\mathbf{C}^t)^T\mathbf{L}^t-(\mathbf{U}^t)^T\mathbf{B}^t\right\|_F^2\\&\quad+\left\|(\mathbf{C}^t)^T\mathbf{L}^t-\mathbf{V}^t\right\|_F^2+\left\|(\mathbf{U}^t)^T\mathbf{B}^t-\mathbf{V}^t\right\|_F^2\\&\quad+\left\|\tanh\left((\mathbf{W}^t)^T\mathbf{X}^t\right)-\mathbf{B}^t\right\|_F^2+\sigma\left(\left\|\mathbf{C}^t\right\|_F^2+\left\|\mathbf{U}^t\right\|_F^2\right)\\&\quad s.t.\ \mathbf{B}^t\in\{-1,+1\}^{k\times n_t}.\end{aligned}\tag{6}$$

Similarly, Eq. (5) is modified as

$$\begin{aligned}&\hat{\mathcal{L}}_2^t(\mathbf{W}^t)\\&=\sum_i\left\|\left(\tanh\left((\mathbf{W}^t)^T\mathbf{S}_{A.i}^t\right)\right)^T\tanh\left((\mathbf{W}^t)^T\mathbf{S}_{B.i}^t\right)-k\right\|_F^2\\&\quad+\sum_i\left\|\left(\tanh\left((\mathbf{W}^t)^T\mathbf{D}_{A.i}^t\right)\right)^T\tanh\left((\mathbf{W}^t)^T\mathbf{D}_{B.i}^t\right)+k\right\|_F^2.\end{aligned}\tag{7}$$

3) Overall Objective Function: Combining Eq. (4) with Eq. (5), we derive the final objective function as follows:

$$\min_{\mathbf{B}^t,\mathbf{C}^t,\mathbf{U}^t,\mathbf{V}^t,\mathbf{W}^t}\mathcal{L}^t=\lambda_1\hat{\mathcal{L}}_1^t+\lambda_2\hat{\mathcal{L}}_2^t,\tag{8}$$

where λ_1 and λ_2 are the parameters to control the importance of $\hat{\mathcal{L}}_1^t$ and $\hat{\mathcal{L}}_2^t$.

C. Alternating Optimization

Apparently, the optimization problem in Eq. (8) is non-convex with the matrices $\mathbf{B}^t$, $\mathbf{C}^t$, $\mathbf{U}^t$, $\mathbf{V}^t$ and $\mathbf{W}^t$, which is thus hard to solve. Fortunately, it is convex with respect to any one of $\mathbf{B}^t$, $\mathbf{C}^t$, $\mathbf{U}^t$, $\mathbf{V}^t$ and $\mathbf{W}^t$ when the other variables are fixed. Therefore, we propose an alternating optimization algorithm to solve the sub-problems with respect to individual variables, as elaborated below.

1) $\mathbf{B}^t$-step: To learn the binary code matrix $\mathbf{B}^t$ by fixing the remaining variables, the sub-optimization of Eq. (8) can be formulated as

$$\begin{aligned}\min_{\mathbf{B}^t} &\left\|(\mathbf{C}^t)^T\mathbf{L}^t-(\mathbf{U}^t)^T\mathbf{B}^t\right\|_F^2+\left\|(\mathbf{U}^t)^T\mathbf{B}^t-\mathbf{V}^t\right\|_F^2\\ &+\left\|\mathbf{B}^t-\tanh\left((\mathbf{W}^t)^T\mathbf{X}^t\right)\right\|_F^2\\ s.t.\ &\mathbf{B}^t\in\{-1,+1\}^{k\times n_t}.\end{aligned}\tag{9}$$

By expanding each term in Eq. (9), we have

$$\begin{aligned}\min_{\mathbf{B}^t}\ &\left\|(\mathbf{C}^t)^T\mathbf{L}^t\right\|_F^2-2\,\mathrm{tr}\left((\mathbf{L}^t)^T\mathbf{C}^t(\mathbf{U}^t)^T\mathbf{B}^t\right)+\left\|(\mathbf{U}^t)^T\mathbf{B}^t\right\|_F^2\\ &+\left\|(\mathbf{U}^t)^T\mathbf{B}^t\right\|_F^2-2\,\mathrm{tr}\left((\mathbf{V}^t)^T(\mathbf{U}^t)^T\mathbf{B}^t\right)+\left\|\mathbf{V}^t\right\|_F^2\\ &+\left\|\mathbf{B}^t\right\|_F^2-2\,\mathrm{tr}\left(\tanh\left((\mathbf{X}^t)^T\mathbf{W}^t\right)\mathbf{B}^t\right)+\left\|\tanh(\mathbf{W}^t)^T\mathbf{X}^t\right\|_F^2\\ s.t.\ &\mathbf{B}^t\in\{-1,+1\}^{k\times n_t},\end{aligned}\tag{10}$$

where $\mathrm{tr}(\mathbf{M})$ denotes the trace of matrix $\mathbf{M}$.

By removing the irrelevant and constant terms, the objective function in Eq. (10) can be rewritten as follows:

$$\begin{aligned}\min_{\mathbf{B}^t}\ &\left\|(\mathbf{U}^t)^T\mathbf{B}^t\right\|_F^2-\mathrm{tr}\left((\mathbf{P}^t)^T\mathbf{B}^t\right)\\ s.t.\ &\mathbf{B}^t\in\{-1,+1\}^{k\times n_t}.\end{aligned}\tag{11}$$

where $\mathbf{P}^t=\mathbf{U}^t(\mathbf{C}^t)^T\mathbf{L}^t+\mathbf{U}^t\mathbf{V}^t+\tanh\left((\mathbf{W}^t)^T\mathbf{X}^t\right)$.

The problem in Eq. (11), however, is NP-hard for directly optimizing binary code matrix $\mathbf{B}^t$. Instead, it can be solved by the method in [9] that updates one row of $\mathbf{B}^t$ while fixing the others each time. Though suboptimal, it is efficient. To elaborate, we first give the following two equations:

$$\begin{aligned}(\mathbf{U}^t)^T\mathbf{B}^t&=(\tilde{\mathbf{u}}_i^t)^T\tilde{\mathbf{b}}_i^t+(\tilde{\mathbf{U}}_i^t)^T\tilde{\mathbf{B}}_i^t\\ s.t.\ &\tilde{\mathbf{b}}_i^t\in\{-1,+1\}^{1\times n_t},\\ &\tilde{\mathbf{B}}_i^t\in\{-1,+1\}^{(k-1)\times n_t},\end{aligned}\tag{12}$$

where $\tilde{\mathbf{b}}_i^t$ and $\tilde{\mathbf{u}}_i^t$ are the i-th rows of $\mathbf{B}^t$ and $\mathbf{U}^t$, respectively. $\tilde{\mathbf{B}}_i^t$ is $\mathbf{B}^t$ excluding $\tilde{\mathbf{b}}_i^t$, and $\tilde{\mathbf{U}}_i^t$ is $\mathbf{U}^t$ excluding $\tilde{\mathbf{u}}_i^t$.

$$(\mathbf{P}^t)^T\mathbf{B}^t=(\tilde{\mathbf{p}}_i^t)^T\tilde{\mathbf{b}}_i^t+(\tilde{\mathbf{P}}_i^t)^T\tilde{\mathbf{B}}_i^t,\tag{13}$$

where $\tilde{\mathbf{p}}_i^t$ is the i-th row of $\mathbf{P}^t$ and $\tilde{\mathbf{P}}_i^t$ is $\mathbf{P}^t$ excluding $\tilde{\mathbf{P}}_i^t$.

Plugging Eq. (12) and Eq. (13) into Eq. (11), we have

$$\begin{aligned}\min_{\tilde{\mathbf{b}}_i^t}&\left\|(\tilde{\mathbf{u}}_i^t)^T\tilde{\mathbf{b}}_i^t\right\|_F^2+\left\|(\tilde{\mathbf{U}}_i^t)^T\tilde{\mathbf{B}}_i^t\right\|_F^2+2\,\mathrm{tr}\left((\tilde{\mathbf{B}}_i^t)^T\tilde{\mathbf{U}}_i^t(\tilde{\mathbf{u}}_i^t)^T\tilde{\mathbf{b}}_i^t\right)\\ &-\mathrm{tr}\left((\tilde{\mathbf{p}}_i^t)^T\tilde{\mathbf{b}}_i^t\right)-\mathrm{tr}\left((\tilde{\mathbf{P}}_i^t)^T\tilde{\mathbf{B}}_i^t\right)\\ s.t.\ &\tilde{\mathbf{b}}_i^t\in\{-1,+1\}^{1\times n_t}.\end{aligned}\tag{14}$$

Note that $\left\|(\tilde{\mathbf{u}}_i^t)^T\tilde{\mathbf{b}}_i^t\right\|_F^2=n_t\left\|\tilde{\mathbf{u}}_i^t\right\|_F^2$ which is a constant. By removing the irrelevant and constant terms, Eq. (14) can be further abbreviated as

$$\begin{aligned}\min_{\tilde{\mathbf{b}}_i^t}&\,\mathrm{tr}\left(2(\tilde{\mathbf{B}}_i^t)^T\tilde{\mathbf{U}}_i^t(\tilde{\mathbf{u}}_i^t)^T\tilde{\mathbf{b}}_i^t-(\tilde{\mathbf{p}}_i^t)^T\tilde{\mathbf{b}}_i^t\right)\\ s.t.\ &\tilde{\mathbf{b}}_i^t\in\{-1,+1\}^{1\times n_t}.\end{aligned}\tag{15}$$

Solving Eq. (15), the i-th row of $\mathbf{B}^t$ can be updated by

$$\tilde{\mathbf{b}}_i^{t+1}=\mathrm{sgn}\left(\tilde{\mathbf{p}}_i^t-2\tilde{\mathbf{u}}_i^t(\tilde{\mathbf{U}}^t)^T\tilde{\mathbf{B}}^t\right),\tag{16}$$

Similarly, the updating formulas of $\mathbf{C}^t$, $\mathbf{U}^t$, $\mathbf{V}^t$ and $\mathbf{W}^t$ can be derived by setting $\frac{\partial\mathcal{L}^t}{\partial\mathbf{C}^t}$, $\frac{\partial\mathcal{L}^t}{\partial\mathbf{U}^t}$, $\frac{\partial\mathcal{L}^t}{\partial\mathbf{V}^t}$, and $\frac{\partial\mathcal{L}^t}{\partial\mathbf{W}^t}$ to zero individually. We then obtain Eqs. (17)–(20), whose detailed derivations are provided in Appendix.

$$\mathbf{C}^{t+1}=\left(2\mathbf{L}^t(\mathbf{L}^t)^T+\sigma\mathbf{I}\right)^{-1}\mathbf{L}^t\left((\mathbf{B}^t)^T\mathbf{U}^t+(\mathbf{V}^t)^T\right),\tag{17}$$

where $\mathbf{I}$ represents the identity matrix.

$$\mathbf{U}^{t+1}=\left(2\mathbf{B}^t(\mathbf{B}^t)^T+\sigma\mathbf{I}\right)^{-1}\mathbf{B}^t\left((\mathbf{L}^t)^T\mathbf{C}^t+(\mathbf{V}^t)^T\right).\tag{18}$$

$$\mathbf{V}^{t+1}=\frac{1}{2}\left((\mathbf{C}^t)^T\mathbf{L}^t+(\mathbf{U}^t)^T\mathbf{B}^t\right)\tag{19}$$

$$\mathbf{W}^{t+1}=\mathbf{W}^t-\eta\frac{\partial\mathcal{L}^t}{\partial\mathbf{W}^t},\tag{20}$$

where η is the learning rate, and $\frac{\partial\mathcal{L}^t}{\partial\mathbf{W}^t}$ is shown in Eq. (24).

We summarize our proposed method in Algorithm 1.

Algorithm 1 Similarity-Preserving Latent Semantic Hashing (SPLH)

Input: Training data set $\mathbf{X}$ with its label space $\mathbf{L}$, the number of hash bits k, the parameters λ, σ, r and η, the total number of stream data batches T.

Output: Binary codes $\mathbf{B}$ for $\mathbf{X}$ and hash weights $\mathbf{W}$.

```
for t = 1 → T do
    Denote the newly coming data chunk as X^t;
    if t = 1 then
        Initialize C^1, U^1, V^1 and W^1 and B^1;
    else
        for i = 1 → k do
            Update B^t by Eq. (16);
        end for
        Update C^t by Eq. (17);
        Update U^t by Eq. (18);
        Update V^t by Eq. (19);
        Update W^t by Eq. (20);
    end if
end for
Set W = W^t;
Compute B = sgn(W^T X);
Return W and B.
```

IV. Time Complexity

The time complexities of updating $\mathbf{B}^t$ in Eq. (16), $\mathbf{C}^t$ in Eq. (17), $\mathbf{U}^t$ in Eq. (18), $\mathbf{V}^t$ in Eq. (19), and $\mathbf{W}^t$ in Eq. (20) are $\mathcal{O}(r^2n_tk)$, $\mathcal{O}(c^3+c^2n_t+n_tkr+n_tcr)$, $\mathcal{O}(k^3+k^2n_t+n_tkr+n_tcr)$, $\mathcal{O}(n_tcr+n_tkr)$ and $\mathcal{O}(n_t^2d^2+n_tdk)$, respectively. Since c, r, k are small values, the computational complexity of each updating depends on the size of a new data chunk and the dimension of features, which is scalable.

TABLE II
PARAMETER SETTINGS OF COMPARED METHODS ON THREE BENCHMARKS

Method	Parameter Tuple	CIFAR-10	Places205	MNIST
OKH	(C, α)	$(0.001, 0.3)$	$(0.0001, 0.7)$	$(0.001, 0.3)$
SketchHash	$(sketch\_size, batch\_size)$	$(200, 50)$	$(100, 50)$	$(200, 50)$
AdaptHash	(α, λ, η)	$(0.9, 0.01, 0.1)$	$(0.9, 0.01, 0.1)$	$(0.8, 0.01, 0.2)$
OSH	η	0.1	0.1	0.1
MIHash	$(\theta, \mathcal{R}, A)$	$(0, 1000, 10)$	$(0, 5000, 10)$	$(0, 1000, 10)$
HCOH	(n_t, η)	$(1, 0.2)$	$(1, 0.1)$	$(1, 0.2)$
BSODH	$(\lambda, \sigma, \eta_s, \eta_d)$	$(0.6, 0.5, 1.2, 0.2)$	$(0.3, 0.5, 1.0, 0.0)$	$(0.9, 0.8, 1.2, 0.2)$

V. EXPERIMENTS

To verify the performance of SPLH, we conduct image retrieval experiments on three widely-used datasets, including CIFAR-10 [58], Places205 [59], and MNIST [60] to compare SPLH with several state-of-the-art methods [21]–[27].

A. Datasets

1) CIFAR-10: CIFAR-10 consists of $60K$ images collected from 10 classes, each containing $6K$ instances where each instance is represented as a 4096-D CNN vector [41]. Similar to [24]–[26], we randomly sample $59K$ instances to form a retrieval set and the remaining $1K$ instances are used as a test set. Besides, we randomly select $20K$ images from the retrieval set as the training set to learn hash functions in a streaming fashion.

*2) Places205: Places*205 [59] is a challenging large-scale dataset containing 2.5 million images collected from 205 scene categories. The features of each image are first extracted from the AlexNet [61] and then reduced to a 128-D feature vector by PCA. Following [25], 20 instances are randomly selected from each category to construct the test set and the remaining is used as the retrieval set. Lastly, a randomly sampled subset of $100K$ images from the retrieval set is used to update the hash functions in a streaming fashion.

3) MNIST: The *MNIST* dataset contains $70K$ handwritten digit images from 0 to 9 [60], where each image is represented by a $28 \times 28 = 784$-D normalized pixel vector. Following the setting in [26], we construct the test set by randomly selecting 100 instances from each class and the rest forms the retrieval set. Finally, a subset of $20K$ instances randomly sampled from the retrieval set is used as the training set to learn the hash functions in a streaming fashion.

B. Evaluation Metrics

Similar to the previous methods, we adopt the following commonly used protocols to evaluate the performance: mean Average Precision (denoted by *mAP*), Precision within a Hamming ball of radius 2 centered at each query (denoted by Precision@H2), mAP vs. different sizes of training instances curves and their corresponding areas under the mAP curves (denoted by mAP-AUC), Precision of the top K retrieved neighbors curves (denoted by Precision@K) and their corresponding areas under the Precision@K curves (denoted by Precision@K-AUC). For each metric, we also compute the improvement gains between the best and second best as $\frac{best-second}{second}$ for comparison.

TABLE III
PARAMETER CONFIGURATIONS OF THE PROPOSED METHOD ON THREE BENCHMARKS

Parameter	CIFAR-10	Places205	MNIST
r	100	50	100
σ	0.1	0.1	0.1
λ_1	0.1	0.1	0.1
λ_2	0.01	0.01	0.01
n_t	100	1000	100

Notably, when reporting the mAP performance on Places205, following the works in [24]–[26], we only compute the top $1,000$ retrieved items (denoted by mAP@1,000 and mAP@1,000-AUC) due to the high complexity of large scale retrieval. The above metrics are evaluated with various hashing code lengths: 8, 16, 32, 48, 64, and 128 bits.

C. Compared Methods and Settings

To verify the effectiveness of the proposed SPLH, we compare our method with several state-of-the-art (STOA) online hashing algorithms including Online Kernel Hashing (OKH) [21], Online Sketching Hashing (SketchHash) [27], Adaptive Hashing (AdaptHash) [22], Online Supervised Hashing (OSH) [23], Online Hashing with Mutual Information (MIHash) [24], Hadamard Codebook based Online Hashing (HCOH) and Balanced Similarity for Online Discrete Hashing (BSODH) [26]. The source codes of these methods are publicly available. Our model is implemented with MATLAB. The experiments are conducted on a server with a 3.60GHz Intel Core I7 4790 CPU and 16G RAM. All experimental results are averaged over three runs. As listed in Table II, we adopt for all three benchmarks the configuration parameters described in [25], [26], that have been carefully validated for each method.

More detailed descriptions of these parameters for each method can be found in [21]–[27]. As for SPLH, we list the parameter configurations for the three benchmarks in Table III. In Sec. V-E, we conduct ablation studies to analyze the best choice of each individual hyper-parameter.

D. Results and Discussions

1) mAP(@1,000) Results: Table IV compares the mAP($@1,000$) performances with various code lengths on CIFAR-10. In addition, the mAP($@1,000$) results on Places205 and MNIST with various code lengths are listed in Table V and Table VI, respectively. The results in these tables show that SPLH apparently outperforms the other baseline methods in most cases. Specifically, on CIFAR-10,

TABLE IV

COMPARISON OF *m*AP AND PRECISION@H2 ON CIFAR-10 WITH VARIOUS CODE LENGTHS OF 8, 16, 32, 48, 64, AND 128 BITS. THE BEST RESULT IS HIGHLIGHTED IN BOLDFACE AND THE SECOND BEST IS UNDERLINED

Method	*m*AP						Precision@H2					
	8-bit	16-bit	32-bit	48-bit	64-bit	128-bit	8-bit	16-bit	32-bit	48-bit	64-bit	128-bit
OKH	0.100	0.134	0.223	0.252	0.268	0.350	0.100	0.175	0.100	0.452	0.175	0.372
SketchHash	0.248	0.301	0.302	0.327	-	-	0.256	0.431	0.385	0.059	-	-
AdaptHash	0.116	0.138	0.216	0.297	0.305	0.293	0.114	0.254	0.185	0.093	0.166	0.164
OSH	0.123	0.126	0.129	0.131	0.127	0.125	0.120	0.123	0.137	0.117	0.083	0.038
MIHash	0.512	0.640	0.675	0.668	0.667	0.664	0.170	0.673	0.657	0.604	0.500	0.413
HCOH	0.536	0.698	0.688	0.707	0.724	0.734	0.333	0.723	0.731	0.694	0.633	0.471
BSODH	0.564	0.604	0.689	0.656	0.709	0.711	0.305	0.582	0.691	0.697	0.690	**0.602**
SPLH	**0.635**	**0.699**	**0.723**	**0.745**	**0.755**	**0.758**	**0.502**	**0.741**	**0.743**	**0.723**	**0.692**	0.566

TABLE V

COMPARISON OF *m*AP@1,000 AND PRECISION@H2 ON PLACES205 WITH VARIOUS CODE LENGTHS OF 8, 16, 32, 48, 64, AND 128 BITS. THE BEST RESULT IS HIGHLIGHTED IN BOLDFACE AND THE SECOND BEST IS UNDERLINED

Method	*m*AP@1,000						Precision@H2					
	8-bit	16-bit	32-bit	48-bit	64-bit	128-bit	8-bit	16-bit	32-bit	48-bit	64-bit	128-bit
OKH	0.018	0.033	0.122	0.048	0.114	0.258	0.007	0.010	0.026	0.017	0.217	0.075
SketchHash	0.052	0.120	0.202	0.242	-	-	0.017	0.066	0.220	0.176	-	-
AdaptHash	0.028	0.097	0.195	0.223	0.222	0.229	0.009	0.051	0.012	0.185	0.021	0.022
OSH	0.018	0.021	0.022	0.032	0.043	0.164	0.007	0.009	0.012	0.023	0.030	0.059
MIHash	0.094	**0.191**	0.244	0.288	0.308	0.332	0.022	0.112	0.204	0.242	0.202	0.069
HCOH	0.049	0.173	0.259	0.280	**0.321**	0.347	0.012	0.082	0.252	0.179	0.114	0.036
BSODH	0.035	0.174	0.250	0.273	0.308	0.337	0.009	0.101	0.241	0.246	0.212	0.101
SPLH	**0.096**	**0.191**	**0.266**	**0.298**	0.320	**0.350**	**0.024**	**0.123**	**0.267**	**0.253**	**0.224**	**0.116**

TABLE VI

COMPARISON OF *m*AP AND PRECISION@H2 ON MNIST WITH VARIOUS CODE LENGTHS OF 8, 16, 32, 48, 64, AND 128 BITS. THE BEST RESULT IS HIGHLIGHTED IN BOLDFACE AND THE SECOND BEST IS UNDERLINED

Method	*m*AP						Precision@H2					
	8-bit	16-bit	32-bit	48-bit	64-bit	128-bit	8-bit	16-bit	32-bit	48-bit	64-bit	128-bit
OKH	0.100	0.155	0.224	0.273	0.301	0.404	0.100	0.220	0.457	0.724	0.522	0.124
SketchHash	0.257	0.312	0.348	0.369	-	-	0.261	0.596	0.691	0.251	-	-
AdaptHash	0.138	0.207	0.319	0.318	0.292	0.208	0.153	0.442	0.535	0.335	0.163	0.168
OSH	0.130	0.144	0.130	0.148	0.146	0.143	0.131	0.146	0.192	0.134	0.109	0.019
MIHash	0.664	0.741	0.744	0.780	0.713	0.681	0.487	0.803	0.814	0.739	0.720	0.471
HCOH	0.536	0.708	0.756	0.772	0.759	0.771	0.350	0.800	0.826	0.766	0.643	0.370
BSODH	0.593	0.700	0.747	0.743	0.766	0.760	0.308	0.709	0.826	0.804	**0.814**	0.643
SPLH	**0.674**	**0.750**	**0.774**	**0.798**	**0.813**	**0.803**	**0.501**	**0.820**	**0.856**	**0.829**	**0.814**	**0.685**

SPLH achieves performance gains over the best baselines, *i.e.*, HCOH or BSODH, by 12.59%, 0.14%, 4.93%, 5.37%, 4.28% and 3.27% corresponding to the code lengths of 8, 16, 32, 48, 64, and 128 bits, respectively. On Places205, SPLH performs the best, except the case of 64-bit code-length, in which SPLH shows slightly poorer performance than HCOH, but is still ranked second.

Concretely, SPLH obtains 2.13%, 9.77%, 2.70%, 3.47% and 0.86% gains over MIHash, HCOH and BSODH corresponding to the code lengths of 8, 16, 32, 48, and 128 bits, respectively. On MNIST, SPLH also outperforms the other methods in all cases. The above results demonstrate that compared with SOTA hashing methods, given a query, SPLH can more accurately rank the relevant items in the top positions.

2) Precision@H2 Results: The Precision@H2 performances evaluated on the three benchmarks are also shown in Tables IV–VI, respectively. Similarly to the results of *m*AP(AP@1, 000) performances, SPLH performs the best in most cases except for the case of 128-bit on CIFAR-10. For example, on CIFAR-10, compared with the SOTAs, *i.e.*, HCOH or BSODH, 50.75%, 2.49%, 1.64%, 3.73% and 0.29% gains are obtained for 8-bit, 16-bit, 32-bit, 48-bit, and 64-bit cases, respectively. Similar performance improvements over SOTAs can also be observed on the Places205 and MNIST datasets. These results again demonstrate the efficacy of SPLH.

Tables IV–VI show an interesting observation that for almost all methods, the Precision@H2 performance is degraded with both low-precision (*i.e.*, 8-bit) and high-precision (*i.e.*, 128-bit) hashing lengths. To explain, a low-precision hash code encodes less semantic information. By contrast, in high-precision code space, the number of hash buckets with a Hamming ball of radius 2 grows rapidly, making the search space too large to learn a code with good precision performance.

3) Influences of Training Data Size: Since SPLH aims to address hash-based retrieval of streaming data, we also show the performances at different updating stages of $\mathbf{W}^t$ and justify the performance gain along with the update of $\mathbf{W}^t$. Without loss of generality, we plot the curves of *m*AP *vs.* different training data size on the three benchmarks in Fig. 3, Fig. 4, and Fig. 5, respectively. On CIFAR-10 and MNIST, the experimental results are performed with the training data size ranges in {2K, 4K, …, 20K}, whereas on Places205, the training data size ranges in {5K, 10K, …, 100K}.

As can be seen, in general, the *m*AP(@1,000) performance for SPLH is better than the others with all different

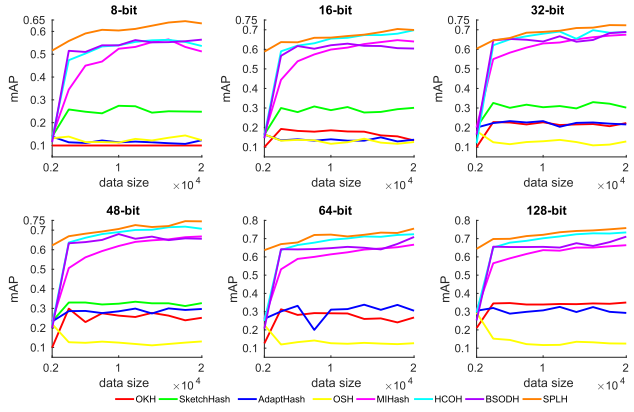


Fig. 3. Comparisons of *m*AP performances with respect to different sizes of training dataset on CIFAR-10.

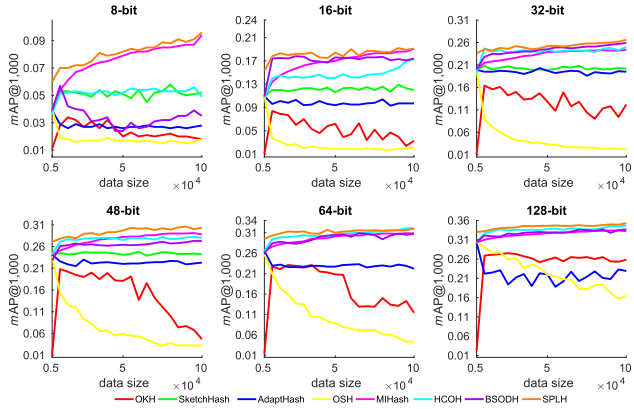


Fig. 4. Comparisons of *m*AP performances with respect to different sizes of training dataset on Places205.

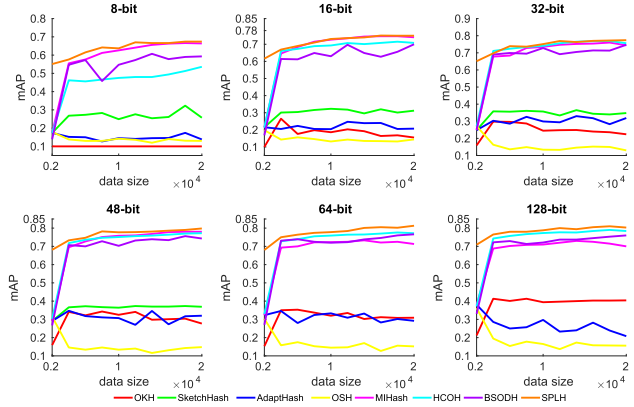


Fig. 5. Comparisons of *m*AP performances with respect to different sizes of training dataset on MNIST.

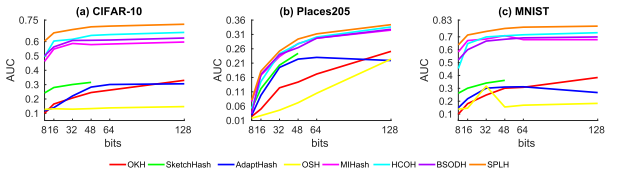


Fig. 6. AUC curves for Fig. 3, Fig. 4 and Fig. 5 on three datasets with different code lengths.

code lengths. To evaluate the robustness of the compared methods, we further show the area under curve (AUC) of the *m*AP(@1,000) *vs.* different training data sizes in Fig. 6.

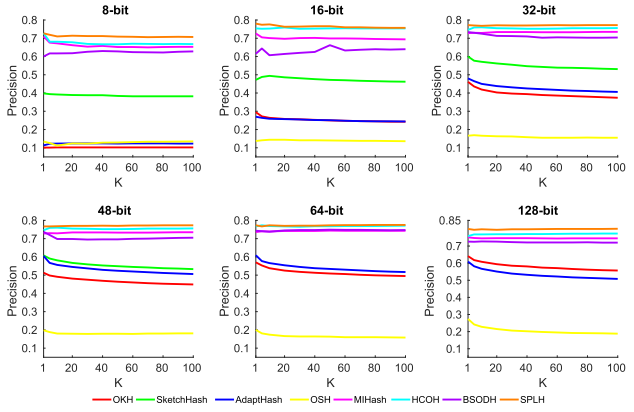


Fig. 7. Precision@K curves with compared algorithms on CIFAR-10.

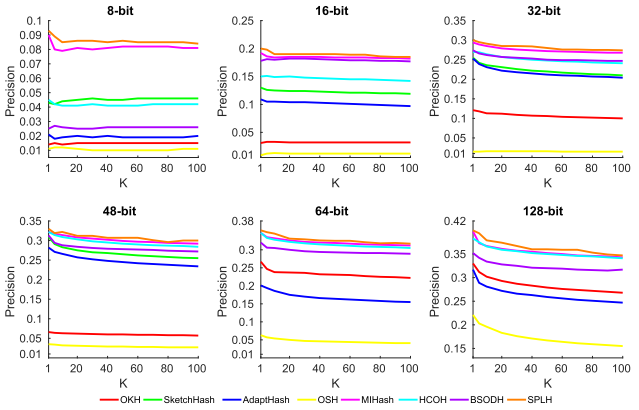


Fig. 8. Precision@K curves with compared algorithms on Places205.

The results show that SPLH achieves the best AUC performances in all cases. Quantitatively, for example, compared with the best SOTA (*i.e.*, HCOH), SPLH shows relative AUC increases of 21.82%, 9.44%, 11.04%, 9.33%, 9.09%, and 8.58% corresponding to the code lengths of 8-bit, 16-bit, 32-bit, 48-bit, 64-bit and 128-bit, respectively, on CIFAR-10. Similarly, general improvements of SPLH over SOTAs on Places205 and MNIST can also be observed as well.

Besides the superior *m*AP(@1,000)-AUC performances, we can also observe another good characteristic of SPLH: the faster adaptivity in online learning compared with SOTAs.

More specifically, in the early training stage as shown in Fig. 3 (data size is 2K), Fig. 4 (data size is 5K) and Fig. 5 (data size is 2K), SPLH achieves a high *m*AP(@1,000) performance with various hash code-lengths. Therefore, SPLH can be fast adapted to online learning, making it particularly suitable for dealing with streaming data.

4) Precision@K Performances: The Precision@*K* performances with various code-lengths on the three benchmarks are illustrated in Fig. 7, Fig. 8, and Fig. 10, respectively. Since for information retrieval, the values of $K \leq 100$ are common choices as they are appropriate to model a user's behavior when investigating search results, we sample the precision results with the value of K falling in $\{1, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$. Besides, the Precision@*K*-AUC results are illustrated in Fig. 11.

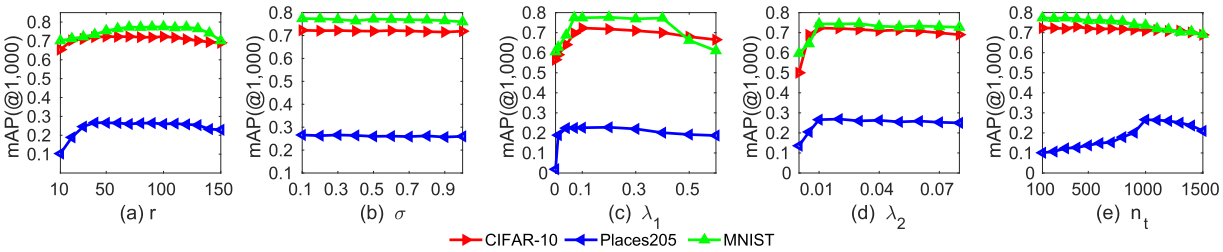


Fig. 9. Comparisons of mAP(@1,000) performances with respect to various values of r, σ, λ_1 λ_2, and n_t with a code length of 32 bits.

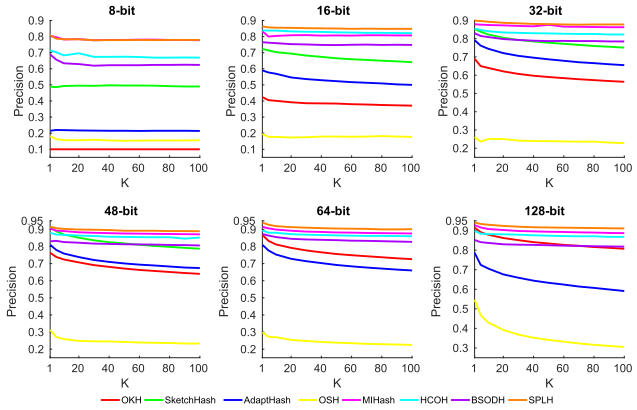


Fig. 10. Precision@K curves with compared algorithms on MNIST.

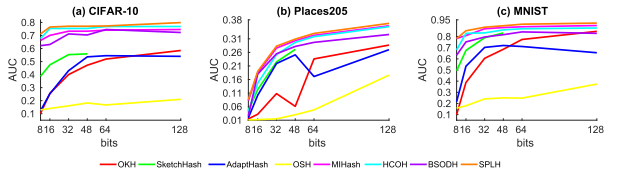


Fig. 11. AUC curves for Fig. 7, Fig. 8 and Fig. 10 on three datasets with various code lengths.

Fig. 7–10 show that the Precision@K curves for SPLH are generally the best among the compared methods, but the improvements over SOTAs on MNIST are less significant. To explain, MNIST is a relatively simple dataset that can be easily well handled by most of the compared methods (The precision ranges in 80% $\sim$ 95% with SOTAs.). Hence, it is hard to boost the performance by a significant margin. Moreover, Fig. 11 also shows that SPLH achieves 5.33%, 1.46%, 2.25%, 2.25%, 0.52% and 3.90% AUC improvements over the best SOTA with various hash code-lengths.

To sum up, the above experimental results and analyses demonstrate the efficacy of SPLH. Moreover, we can observe that latent linkage space based SPLH outperforms the other supervised frameworks based on discrete labels or pairwise similarity, which justifies the correctness and importance of the proposed latent linkage space.

E. Ablation Study

In this section, we conduct the ablation studies on the hyper-parameters used in our method, including the dimension of latent semantic space r, the regularization parameter σ, the trade-off parameters λ_1 and λ_2, and the batch size of streaming data n_t. To that effect, we conduct experiments with various values of these hyper-parameters *w.r.t* mAP(@1,000) in the case of 32-bit in Fig. 9 (Detailed parameter settings used in this paper are listed in Table III.).

1) Influence of r: We first start with the analysis on the dimension r of latent semantic space. As shown in Fig. 9(a), when $r < 60$, the mAP(@1,000) on CIFAR-10 and MNIST increases with r. When $r \in [60, 120]$, the performance becomes stable. When $r > 120$, the mAP(@1,000) gets degraded. Similarly, the stable interval on Places205 falls in [40, 100]. This well demonstrates our assumption on the existence of the linkage space and such a linkage space exists in a particular space dimension for each dataset. Based on the above results, we set the value of r as 100, 50 and 100 for CIFAR-10, Places205 and MNIST, respectively, in our experiments.

2) Influence of σ: Fig. 9(b) depicts the mAP(@1,000) performance with various values of σ, showing that SPLH is insensitive to the values of σ. To analyze, σ is used to guarantee the reversibility of Eq. (17) and Eq. (18). Hence, it does not affect the performance of SPLH significantly. In our experiments, we set σ as 0.1 for all three benchmarks.

3) Influences of λ_1 and λ_2: λ_1 controls the importance of the loss term in Eq. (6) that is related to the latent linkage space, whereas λ_2 controls the importance of the similarity-preserving loss in Eq. (7). Fig. 9(c) shows that the value of λ_1 does affect the performance of SPLH. When $\lambda_1 = 0.1$, SPLH achieves the best mAP(@1,000) performances of 0.723, 0.266 and 0.774 on CIFAR-10, Places205 and MNIST, respectively. However, when $\lambda_1 = 0$, i.e., overlooking the linkage space, the performance of SPLH is significantly degraded to 0.565, 0.136 and 0.605 on the three datasets, respectively. We can observe from Fig. 9(c) that, properly applying the loss term of the linkage space in Eq. (6) can significantly boost the performance by 27.96%, 95.59% and 27.93% on CIFAR-10, Places205 and MNIST, respectively, verifying the effectiveness and correctness of learning the latent linkage space.

Fig. 9(d) shows that when $\lambda_2 = 0.01$, the mAP(@1,000) performance reaches the best results of 0.723, 0.266 and 0.774 on CIFAR-10, Places205 and MNIST, respectively, and then starts to decrease when λ_2 further increases. When $\lambda_2 = 0$, i.e., overlooking the effectiveness of similarity-preserving loss, the performance of SPLH also significantly decreases to 0.500, 0.136, and 0.597 on the three datasets, respectively. Thus, exploring similarity-preservation is also important in learning effective binary codes.

To sum up, to obtain satisfactory online retrieval performance, both the latent semantic space learning and similarity

TABLE VII
TRAINING TIME ON THREE BENCHMARKS UNDER 32-BIT HASHING CODES

Method	CIFAR-10 (s)	Places205 (s)	MNIST (s)
OKH	4.53	15.66	4.58
SketchHash	4.98	3.52	1.27
AdaptHash	20.73	14.49	6.26
OSH	93.45	56.68	24.07
MIHash	120.10	468.77	97.59
HCOH	12.34	10.54	4.01
BSODH	36.12	69.73	4.83
SPLH	28.91	17.33	4.32

preservation matter. In the experiments, we set the value of tuple (λ_1, λ_2) as (0.1, 0.01).

4) Influence of n_t: The experimental results *w.r.t* the batch size of streaming data n_t ranging in $\{100, 200, \ldots, 1500\}$ are illustrated in Fig. 9(e). As we can see, on CIFAR-10 and MNIST, the performance of SPLH decreases with n_t, whereas on Places205, the performance of SPLH increases until $n_t \approx 1K$. In the experiments, we set n_t as 100 for CIFAR-10 and MNIST, and $1K$ for Places205. Such setting conforms with online streaming data since $n_t \ll n$ in our experiments, when n is the number of training samples ($n = 20K$ for CIFAR-10 and MNIST and $n = 100K$ for Places205).

F. Training Efficiency

To further evaluate the training efficiency of SPLH, we conduct experiments given hashing bit $r = 32$. Similar observations below can be found with other code lengths. The time consumption of each method is shown in Tab. VII.

Generally, OKH and SketchHash are the most efficient, which however suffer poor *m*AP (*m*AP@1,000) as shown in Fig. 3, Fig. 4 and Fig. 5. Besides, AdaptHash and HCOH are also computationally more efficient than the proposed SPLH. However, similar to OKH and SketchHash, AdaptHash shows poor performance. And HCOH performs significantly worse with low code lengths, *e.g.*, 8-bit and 16-bit as shown in Tab.IV, Tab. V and Tab. VI. Compared with state-of-the-art methods, MIHash and BSODH, SPLH shows consistently better efficiency. As analyzed in Sec. II, MIHash has to calculate the Hamming distance between the neighbors and non-neighbors for each instance. As for BSODH, the discrete optimization adopted in BSODH brings about more variables and the convergence problem. Besides, it requires a large data batch to update the hash functions at each round. It can be observed that SPLH consumes more training time on CIFAR-10 and Places205 than on MNIST. We argue that this is owing to the high feature dimension of CIFAR-10 (4096-dim) and large scale of Places205 (100, 000). Nevertheless, the proposed SPLH can keep a good balance between the effectiveness and efficiency, thereby making it scalable to applications of various scales.

VI. CONCLUSION

In this work, we proposed a similarity-preserving linkage hashing scheme for online image retrieval. Our method aims to learn a latent linkage space where the relationships for different semantic concepts are well captured while preserving the semantic similarity between neighboring data points. To this end, our method first transforms independent discrete labels and binary codes into a latent linkage space, through which the relative semantic distances between data points can be evaluated more precisely. Then, pairwise similarity is used towards more robust binary codes to guide the learning of the latent linkage space. We have also derived an iterative alternating optimization approach to efficiently solve the proposed model. Extensive experimental results on three widely-adopted datasets demonstrate the effectiveness of the proposed approach.

APPENDIX

We show the detailed derivations of the updating formulas in Eqs. (17)–(20).

1) $\mathbf{C}^t$-step: Fixing $\mathbf{B}^t$, $\mathbf{U}^t$, $\mathbf{V}^t$ and $\mathbf{W}^t$, and setting the partial derivative of Eq. (8) *w.r.t.* $\mathbf{C}^t$ to zero, we have

$$\frac{\partial \mathcal{L}^t}{\partial \mathbf{C}^t} = \left(2\mathbf{L}^t(\mathbf{L}^t)^T + \sigma\mathbf{I}\right)\mathbf{C}^t - \mathbf{L}^t\left((\mathbf{B}^t)^T\mathbf{U}^t + (\mathbf{V}^t)^T\right) = \mathbf{0}. \tag{21}$$

Then, we can obtain the closed-form solution of $\mathbf{C}^t$ as in Eq. (17).

2) $\mathbf{U}^t$-step: Fixing $\mathbf{B}^t$, $\mathbf{C}^t$, $\mathbf{V}^t$ and $\mathbf{W}^t$, and setting the partial derivative of Eq. (8) *w.r.t.* $\mathbf{U}^t$ to zero, we have

$$\frac{\partial \mathcal{L}^t}{\partial \mathbf{U}^t} = \left(2\mathbf{B}^t(\mathbf{B}^t)^T + \sigma\mathbf{I}\right)^{-1}\mathbf{U}^t - \mathbf{B}^t\left((\mathbf{L}^t)^T\mathbf{C}^t + \mathbf{V}^{t^T}\right) = \mathbf{0}. \tag{22}$$

Then, we can obtain the closed-form solution of $\mathbf{U}^t$ as in Eq. (18).

3) $\mathbf{V}^t$-Step: Fixing $\mathbf{B}^t$, $\mathbf{C}^t$, $\mathbf{U}^t$ and $\mathbf{W}^t$, and setting the partial derivative of Eq. (8) *w.r.t.* $\mathbf{V}^t$ to zero, we have

$$\frac{\partial \mathcal{L}}{\partial \mathbf{V}^t} = 2\mathbf{V}^t - (\mathbf{C}^t)^T\mathbf{L}^t - (\mathbf{U}^t)^T\mathbf{B}^t = \mathbf{0}. \tag{23}$$

Then, we can obtain the closed-form solution of $\mathbf{V}^t$ as in Eq. (19).

4) $\mathbf{W}^t$-Step: Fixing $\mathbf{B}^t$, $\mathbf{C}^t$, $\mathbf{U}^t$ and $\mathbf{V}^t$, and we obtain the derivative of Eq. (8) *w.r.t.* $\mathbf{W}^t$ as

$$\frac{\partial \mathcal{L}^t}{\partial \mathbf{W}^t} = \frac{\partial \hat{\mathcal{L}}_1^t}{\mathbf{W}^t} + \lambda\frac{\hat{\mathcal{L}}_2^t}{\mathbf{W}^t}, \tag{24}$$

where

$$\frac{\partial \hat{\mathcal{L}}_1^t}{\mathbf{W}^t} = -\mathbf{X}^t\Bigg(\Big(\mathbf{B}^t - \tanh\big((\mathbf{W}^t)^T\mathbf{X}^t\big)\Big) \odot \Big(1 - \tanh\big((\mathbf{W}^t)^T\mathbf{X}^t\big)\odot\tanh\big((\mathbf{W}^t)^T\mathbf{X}^t\big)\Big)\Bigg), \tag{25}$$

and

$$\frac{\hat{\mathcal{L}}_2^t}{\mathbf{W}^t} = \mathbf{S}_A^t(\mathbf{T}_S^t \odot \mathbf{T}_{SA}^t) + \mathbf{S}_B^t(\mathbf{T}_S^t \odot \mathbf{T}_{SB}^t) + \mathbf{D}_A^t(\mathbf{T}_D^t \odot \mathbf{T}_{DA}^t) + \mathbf{D}_B^t(\mathbf{T}_D^t \odot \mathbf{T}_{DB}^t), \tag{26}$$

where

$$(\mathbf{T}_S^t)_{ij} = \left((\mathbf{S}_A^t)_{.i}\right)^T(\mathbf{S}_B^t)_{.i}, \tag{27}$$

[1]

and

$$\mathbf{T}_{SA}^{t} = \left(1 - \tanh\left((\mathbf{S}_{A}^{t})^{T}\mathbf{W}^{t}\right) \odot \tanh\left((\mathbf{S}_{A}^{t})^{T}\mathbf{W}^{t}\right)\right) \odot \tanh\left((\mathbf{S}_{B}^{t})^{T}\mathbf{W}^{t}\right), \quad (28)$$

$$\mathbf{T}_{SB}^{t} = \left(1 - \tanh\left((\mathbf{S}_{B}^{t})^{T}\mathbf{W}^{t}\right) \odot \tanh\left((\mathbf{S}_{B}^{t})^{T}\mathbf{W}^{t}\right)\right) \odot \tanh\left((\mathbf{S}_{A}^{t})^{T}\mathbf{W}^{t}\right), \quad (29)$$

where

$$(\mathbf{T}_{D}^{t})_{ij} = \left((\mathbf{D}_{A}^{t})_{.i}\right)^{T}(\mathbf{D}_{B}^{t})_{.i}, ^{2} \quad (30)$$

$$\mathbf{T}_{DA}^{t} = \left(1 - \tanh\left((\mathbf{D}_{A}^{t})^{T}\mathbf{W}^{t}\right) \odot \tanh\left((\mathbf{D}_{A}^{t})^{T}\mathbf{W}^{t}\right)\right) \odot \tanh\left((\mathbf{D}_{B}^{t})^{T}\mathbf{W}^{t}\right), \quad (31)$$

and

$$\mathbf{T}_{DB}^{t} = \left(1 - \tanh\left((\mathbf{D}_{B}^{t})^{T}\mathbf{W}^{t}\right) \odot \tanh\left((\mathbf{D}_{B}^{t})^{T}\mathbf{W}^{t}\right)\right) \odot \tanh\left((\mathbf{D}_{A}^{t})^{T}\mathbf{W}^{t}\right), \quad (32)$$

where $\odot$ denotes the Hadamard product.

Then, we update $\mathbf{W}^t$ through online stochastic gradient descent (OSGD) as in Eq. (20).

Consequently, the objective function $\mathcal{L}^t$ can be minimized by updating $\mathbf{B}^t$, $\mathbf{C}^t$, $\mathbf{U}^t$, $\mathbf{V}^t$, and $\mathbf{W}^t$ step-by-step. For round $t = 1$, $\mathbf{C}^1$, $\mathbf{U}^1$, $\mathbf{V}^1$, and $\mathbf{W}^1$ are initialized with matrices containing normally distributed random numbers. As for $\mathbf{B}^1$, we initialize it with $\mathrm{sgn}\left((\mathbf{W}^1)^T\mathbf{X}^1\right)$.

REFERENCES

[1] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. NIPS*, 2009, pp. 1753–1760.

[2] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3424–3431.

[3] S. Kumar and R. Udupa, "Learning hash functions for cross-view similarity search," in *Proc. IJCAI*, 2011, pp. 1–6.

[4] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[5] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2074–2081.

[6] K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2938–2945.

[7] M. Rastegari, J. Choi, S. Fakhraei, D. Hal, and L. Davis, "Predictable dual-view hashing," in *Proc. ICML*, 2013, pp. 1328–1336.

[8] X. Zhu, L. Zhang, and Z. Huang, "A sparse embedding and least variance encoding approach to hashing," *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 3737–3750, Sep. 2014.

[9] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 37–45.

[10] L. Liu, M. Yu, and L. Shao, "Multiview alignment hashing for efficient image search," *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 956–966, Mar. 2015.

[11] J. Tang, K. Wang, and L. Shao, "Supervised matrix factorization hashing for cross-modal retrieval," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3157–3166, Jul. 2016.

[12] F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen, and D. Tao, "A fast optimization method for general binary code learning," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5610–5621, Dec. 2016.

[13] J. Lu, G. Wang, and J. Zhou, "Simultaneous feature and dictionary learning for image set based face recognition," *IEEE Trans. Image Process.*, vol. 26, no. 8, pp. 4042–4054, Aug. 2017.

[14] J. Lu, J. Hu, and Y.-P. Tan, "Discriminative deep metric learning for face and kinship verification," *IEEE Trans. Image Process.*, vol. 26, no. 9, pp. 4269–4282, Sep. 2017.

[15] L. Jin, K. Li, Z. Li, F. Xiao, G.-J. Qi, and J. Tang, "Deep semantic-preserving ordinal hashing for cross-modal similarity search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1429–1440, May 2019.

[16] Y. Duan, J. Lu, J. Feng, and J. Zhou, "Context-aware local binary feature learning for face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 5, pp. 1139–1153, May 2018.

[17] H. Liu, M. Lin, S. Zhang, Y. Wu, F. Huang, and R. Ji, "Dense auto-encoder hashing for robust cross-modality retrieval," in *Proc. ACM Multimedia Conf. Multimedia Conf. (MM)*, 2018, pp. 1589–1597.

[18] X. Liu, X. Nie, W. Zeng, C. Cui, L. Zhu, and Y. Yin, "Fast discrete cross-modal hashing with regressing from semantic labels," in *Proc. ACM Multimedia Conf. (MM)*, 2018, pp. 1662–1669.

[19] Y. Shen, L. Liu, and L. Shao, "Unsupervised binary representation learning with deep variational networks," *Int. J. Comput. Vis.*, vol. 127, nos. 11–12, pp. 1614–1628, Dec. 2019.

[20] Y. Duan, J. Lu, Z. Wang, J. Feng, and J. Zhou, "Learning deep binary descriptor with multi-quantization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 8, pp. 1924–1938, Aug. 2019.

[21] L.-K. Huang, Q. Yang, and W.-S. Zheng, "Online hashing," in *Proc. IJCAI*, 2013, pp. 1–7.

[22] F. Cakir and S. Sclaroff, "Adaptive hashing for fast similarity search," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 1044–1052.

[23] F. Cakir, S. A. Bargal, and S. Sclaroff, "Online supervised hashing," *Comput. Vis. Image Understand.*, vol. 156, pp. 162–173, Mar. 2017.

[24] F. Cakir, K. He, S. A. Bargal, and S. Sclaroff, "MIHash: Online hashing with mutual information," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 437–445.

[25] M. Lin, R. Ji, H. Liu, and Y. Wu, "Supervised online hashing via Hadamard codebook learning," in *Proc. ACM Multimedia Conf. (MM)*, 2018, pp. 1635–1643.

[26] M. Lin, R. Ji, H. Liu, X. Sun, Y. Wu, and Y. Wu, "Towards optimal discrete online hashing with balanced similarity," in *Proc. AAAI Conf. Artif. Intell.*, Jul. 2019, pp. 8722–8729.

[27] C. Leng, J. Wu, J. Cheng, X. Bai, and H. Lu, "Online sketching hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2503–2511.

[28] X. Chen, I. King, and M. R. Lyu, "FROSH: Faster online sketching hashing," in *Proc. UAI*, 2017, pp. 1–10.

[29] S. J. Pan, I. W. Tsang, J. T. Kwok, and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 2, pp. 199–210, Feb. 2011.

[30] Y. Yang, Y. Luo, W. Chen, F. Shen, J. Shao, and H. T. Shen, "Zero-shot hashing via transferring supervised knowledge," in *Proc. ACM Multimedia Conf. (MM)*, 2016, pp. 1286–1295.

[31] H. Zhang, Y. Long, and L. Shao, "Zero-shot hashing with orthogonal projection for image retrieval," *Pattern Recognit. Lett.*, vol. 117, pp. 201–209, Jan. 2019.

[32] K. Ding, C. Huo, B. Fan, S. Xiang, and C. Pan, "In defense of locality-sensitive hashing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 87–103, Jan. 2018.

[33] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. FOCS*, 2006, pp. 459–468.

[34] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. 20th Annu. Symp. Comput. Geometry (SCG)*, 2004, pp. 253–262.

[35] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. NIPS*, 2009, pp. 1509–1517.

[36] Y. Mu and S. Yan, "Non-metric locality-sensitive hashing," in *Proc. AAAI*, 2010, pp. 1–10.

[37] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. ICML*, 2011, pp. 1–8.

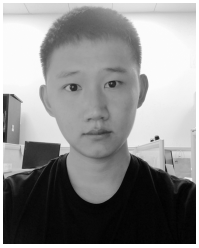
[38] Q.-Y. Jiang and W.-J. Li, "Scalable graph hashing with feature transformation," in *Proc. AAAI*, 2015, pp. 1–7.

[39] H. Liu, R. Ji, J. Wang, and C. Shen, "Ordinal constraint binary coding for approximate nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 4, pp. 941–955, Apr. 2019.

[40] X. Liu, X. Nie, Q. Zhou, L. Nie, and Y. Yin, "Model optimization boosting framework for linear model hash learning," *IEEE Trans. Image Process.*, vol. 29, pp. 4254–4268, 2020.

[41] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015, pp. 1–14.

[42] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.
[43] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.
[44] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI*, 2014, pp. 1–7.
[45] J. Lu, V. E. Liong, and J. Zhou, "Deep hashing for scalable image search," *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2352–2367, May 2017.
[46] Y. Cao, M. Long, B. Liu, and J. Wang, "Deep Cauchy hashing for Hamming space retrieval," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1229–1237.
[47] E. Yang, T. Liu, C. Deng, W. Liu, and D. Tao, "DistillHash: Unsupervised deep hashing by distilling data pairs," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2946–2955.
[48] S. Jin, H. Yao, X. Sun, S. Zhou, L. Zhang, and X. Hua, "Deep saliency hashing for fine-grained retrieval," *IEEE Trans. Image Process.*, to be published.
[49] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *J. Mach. Learn. Res.*, vol. 7, pp. 551–585, Mar. 2006.
[50] J. Jiang and Z. Tu, "Efficient scale space auto-context for image segmentation and labeling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 1810–1817.
[51] J. Kittler, R. Ghaderi, T. Windeatt, and J. Matas, "Face verification using error correcting output codes," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2001, pp. 1–6.
[52] R. E. Schapire, "Using output codes to boost multiclass learning problems," in *Proc. ICML*, 1997, pp. 313–321.
[53] K. J. Horadam, *Hadamard Matrices and Their Applications*. Princeton, NJ, USA: Princeton Univ. Press, 2012.
[54] M. Lin, R. Ji, H. Liu, X. Sun, S. Chen, and Q. Tian, "Hadamard matrix guided online hashing," 2019, *arXiv:1905.04454*. [Online]. Available: http://arxiv.org/abs/1905.04454
[55] Z. Weng and Y. Zhu, "Online hashing with efficient updating of binary codes," in *Proc. AAAI*, 2020, pp. 1–9.
[56] E. Liberty, "Simple and deterministic matrix sketching," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2013, pp. 581–588.
[57] Q. Jiang and W. Li, "Asymmetric deep supervised hashing," in *Proc. AAAI*, 2018, pp. 1–8.
[58] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. 1, 2009.
[59] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Proc. NIPS*, 2014, pp. 487–495.
[60] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[61] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

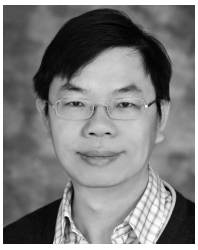


Mingbao Lin received the M.S. degree from Xiamen University, China, in 2018, where he is currently pursuing the Ph.D. degree. His research interests include information retrieval and computer vision.

Rongrong Ji (Senior Member, IEEE) is currently a Professor and the Director of the Intelligent Multimedia Technology Laboratory, and the Dean Assistant with the School of Information Science and Engineering, Xiamen University, Xiamen, China. His work mainly focuses on innovative technologies for multimedia signal processing, computer vision, and pattern recognition, with over 100 papers published in international journals and conferences. He is a member of the ACM. He also serves as a program committee member for several Tier-1 international conferences. He was a recipient of the ACM Multimedia Best Paper Award and the Best Thesis Award of Harbin Institute of Technology. He serves as an Associate/Guest Editor for international journals and magazines, such as *Neurocomputing*, *Signal Processing*, *Multimedia Tools and Applications*, the *IEEE Multimedia Magazine*, and *Multimedia Systems*.

Shen Chen received the bachelor's degree in computer science from Fuzhou University, China, in 2018. He is currently pursuing the master's degree with the MAC Laboratory, Xiamen University. His research interests include hash retrieval and face recognition.

Xiaoshuai Sun received the B.S. degree in computer science from Harbin Engineering University, Harbin, China, in 2007, and the M.S. and Ph.D. degrees in computer science and technology from the Harbin Institute of Technology, Harbin, in 2009 and 2015, respectively. He was a Postdoctoral Research Fellow with the University of Queensland from 2015 to 2016. He served as a Lecturer with the Harbin Institute of Technology from 2016 to 2018. He is currently an Associate Professor with Xiamen University, China. He was a recipient of the Microsoft Research Asia Fellowship in 2011.



Chia-Wen Lin (Fellow, IEEE) received the Ph.D. degree in electrical engineering from National Tsing Hua University (NTHU), Hsinchu, Taiwan, in 2000.

He was with the Department of Computer Science and Information Engineering, National Chung Cheng University, Taiwan, from 2000 to 2007. Prior to joining academia, he worked for the Information and Communications Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan, from 1992 to 2000. He is currently a Professor with the Department of Electrical Engineering and the Institute of Communications Engineering, NTHU. He is also the Deputy Director of the AI Research Center, NTHU. His research interests include image and video processing, computer vision, and video networking.

Dr. Lin was a Steering Committee member of the IEEE TRANSACTIONS ON MULTIMEDIA from 2014 to 2015. He was a Distinguished Lecturer of the IEEE Circuits and Systems Society from 2018 to 2019. He has also served as the President of the Chinese Image Processing and Pattern Recognition Association, Taiwan, from 2010 to 2019. His papers won the Best Paper Award of IEEE VCIP 2015, Top 10% Paper Awards of IEEE MMSP 2013, and the Young Investigator Award of VCIP 2005. He received the Young Investigator Award presented by Ministry of Science and Technology, Taiwan, in 2006. He was the Chair of the Multimedia Systems and Applications Technical Committee of the IEEE Circuits and Systems Society from 2013 to 2015. He has served as the Technical Program Co-Chair of IEEE ICME 2010. He will be the General Co-Chair of IEEE VCIP 2018 and Technical Program Co-Chair of IEEE ICIP 2019. He has served as an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE MULTIMEDIA, and the *Journal of Visual Communication and Image Representation*.