Incremental Hash-bit Learning for Semantic Image Retrieval in Non-stationary Environments

Wing W.Y. Ng, *Senior Member, IEEE,* Xing Tian*, Witold Pedrycz, *Fellow, IEEE,* Xizhao Wang, *Fellow, IEEE,* and Daniel S. Yeung, *Fellow, IEEE*

***Abstract*—Images are uploaded to the Internet over time which makes concept drifting and distribution change in semantic classes unavoidable. Current hashing methods being trained using a given static database may not be suitable for non-stationary semantic image retrieval problems. Moreover, directly re-training a whole hash table to update knowledge coming from new arriving image data may not be efficient. Therefore, this work proposes a new incremental hash-bit learning method. At the arrival of new data, hash bits are selected from both existing and newly trained hash bits by an iterative maximization of a 3-component objective function. This objective function is also used to weight selected hash bits to re-rank retrieved images for better semantic image retrieval results. The three components evaluate a hash bit in three different angles: information preservation, partition balancing, and bit angular difference. The proposed method combines knowledge preserved from previously trained hash bits and new semantic knowledge learned from the new data by training new hash bits. In comparison to table-based incremental hashing, the proposed method automatically adjusts the number of bits from old data and new data according to the concept drifting in the given data via the maximization of the objective function. Experimental results show that the proposed method outperforms existing stationary hashing methods, table-based incremental hashing, and online hashing methods in 15 different simulated non-stationary data environments.**

***Index Terms*—Hashing, Non-stationary Environment, Image Retrieval, Concept Drift, Hash Bit Learning.**

I. Introduction

With the increase in the number of cameras and smartphones being widely used in daily life, the number of images available on the Internet grows explosively every day. For instance, Flickr has announced that 6 billion images are hosted and 3.5 million images are uploaded daily. Newly added images change the data distribution of semantic classes and therefore lead to the occurrence of concept drift problems. Some of those millions of newly added images may present as a new concept, e.g. new ideas like ice bucket challenge or posters of new movies, while some may change the distribution of an existing semantic class, e.g. a new model of car being released. Even those images may not add dramatic changes to the data distribution of images every time, gradual and accumulated changes of data distribution may lead to the failure of current fixed database hashing-based image retrieval methods.

Tree-based retrieval methods, e.g. kd-tree [1] and cover trees [2] are instances of early methods used to solve the nearest neighbor search problems for image retrieval. In ideal cases, theoretically, tree-based methods return accurate retrieval results in sub-linear time complexity. However, the performance of tree-based methods drops significantly when the dimensionality of features is high. Moreover, the storage of tree structures is memory demanding which makes them impractical for large scale databases. Hashing methods are approximated nearest neighbor search methods with sub-linear complexity which project images from a high dimensional feature space onto a low dimensional Hamming space to generate binary hash codes for images. Hash functions partition the data feature space into many hash buckets where images in the same hash bucket share the same hash code. Finally, the similarity between images is evaluated by calculating the Hamming distance between codes of images rather than the Euclidian distance between their feature vectors.

However, most of existing hashing methods assume that all images are available in advance and the data distribution will not change. In fact, the data environment on the Internet is always non-stationary because new images are uploaded over time. Meanwhile, the distribution of semantic classes may drift, i.e., concept drift as aforementioned [3]. Therefore, semantic image retrieval problems are unavoidably non-stationary with concept drift omnipresent in practical environments. Both the Online Kernel-based Hashing (OKH) [4] and the Online Sketching Hashing (OSH) [5] assume new data appears in online manner but ignoring the concept drift problems. The Incremental Hashing (ICH) [6] is proposed as the first work to deal with the concept drift problem in non-stationary data environments. However, employing multiple hash tables and re-weight ensemble of hash tables from different data batches may create duplicated hash bits and reduce retrieval efficiency.

Therefore, the Incremental hash-Bit Learning (IBL) method is proposed in this paper to train and select hash bits when new data arrives via a maximization of a 3-component objective function. The bit learning procedures in the IBL consist of bit training, bit weighting, and bit selection steps. In the IBL, both

Wing W.Y. Ng (wingng@ieee.org) and Xing Tian (shawntian123@gmail.com, corresponding author) are with the Guangdong Provincial Key Laboratory of Computational Intelligence and Cyberspace Information, School of Computer Science and Engineering, South China University of Technology, Guangzhou, Guangdong, 510006, China.

Daniel S. Yeung is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou, Guangdong, 510006, China.

Witold Pedrycz is with the Department of Electrical and Computer Engineering, University of Alberta, Canada, also with the Department of Electrical and Computer Engineering Faculty of Engineering, King Abdulaziz University Jeddah, 21589, Saudi Arabia, and also with the Systems Research Institute, Polish Academy of Sciences Warsaw, Poland.

Xizhao Wang is with the College of Computer Science and Software Engineering, Shenzhen University, China.

TABLE I: List of symbols

i, j, k	Index variables	w_k	The projection vector of the k^{th} hash function
x	An image sample	a_k	The intercept vector of the k^{th} hash function
γ	Similarity consistency in the ICH	$sgn()$	The sign function
δ	Code variance in the ICH	ψ	The hash function pool
r_i	Weight for the i^{th} hash table in the ICH	υ_k	The weight of the k^{th} hash function in ψ
Q	Number of images for return in the ICH	Ψ	The selected hash function set
$f_i(x_j, x_k)$	The Hamming distance between x_j and x_k for the i^{th} hash table in the ICH	v_k	The weight of the k^{th} hash function in Ψ
φ_+	Similar image pairs set	V	The weight vector for hash functions in Ψ
φ_-	Dissimilar image pairs set	$H(x_i, x_j)$	The weighted Hamming distance between x_i and x_j
σ_k	The variance for the k^{th} hash bit	$P(h_k)$	Information preservation of h_k
K	The number of bits in a hash table in the ICH	$B(h_k)$	Partition balancing of h_k
t	Time step	$I(h_k, h_i)$	Bit angular difference between h_k and h_i
D_T	The data chunk arrives at time T	X_L	The set of labeled images in the data chunk
n	The number of images in a data chunk	L	The number of labeled images in X_L
d	Dimensionality of the vector describing an image	S	The pairwise similarity matrix
b	Number of hash functions (bits) being trained in each time step	y_k	The hash function value vector for all labeled images based on the k^{th} hash function
h_k	The k^{th} hash function	Y_k	The hash function value vector for all images based on the k^{th} hash function
W_T	The set of hash functions trained at time T	m	The multiple of hash bits used in the light version of IBL

existing and newly trained hash functions are stored in a hash function pool to preserve knowledge learned from both old data chunks and the newly arrived data chunk. The weighting scheme of the IBL selects the set of hash functions which are the most suitable to the current updated data environment. Selected bits are also weighed by the same objective function to provide a weighed Hamming distance for image similarity computation to enhance the retrieval results. Commonly used symbols are listed in Table I.

The motivation behind the proposal of IBL is that existing hashing methods proposed for non-stationary environments update an entire hash table (a set of hash bits) in iteration which may create redundant hash functions across different tables and yield lower efficiency. Thus, the IBL is proposed to learn at the bit-level instead of hash tables. To our best knowledge, the IBL is the first bit learning method for non-stationary semantic image retrieval. Moreover, the non-stationary retrieval problem is newly proposed in [6] which is different from incremental learning problems. Incremental learning problems only focus on the classification or regression on the most updated newly arrived samples while the non-stationary retrieval problem retrieves image (or information) from the entire database without regarding the time of arrival for a given query from the newly updated distribution. Moreover, a 3-component objective function is proposed to evaluate a hash bit in three different angles for optimal hash bit selection.

The major contributions of this paper are as follows:

- The IBL is the first semi-supervised hash bit learning method for solving non-stationary semantic image retrieval problems with concept drift. In contrast to the ICH, the IBL removes duplicated bits for better retrieval efficiency. The IBL is adaptive to different non-stationary data environments with concept drift.
- A bit weighting scheme is proposed in this paper. The objective function of the IBL evaluates a hash bit in three different aspects: information preservation, partition balancing, and bit angular difference. The objective function values for hash bits are used as weights for both hash bit selection in training and re-ranking in retrieval.
- A light version of the IBL (IBL/L) is proposed which reduces the number of hash bits for light storage while achieves similar performance with the original IBL.

The paper is organized as follows. Section II introduces related works on representative hashing methods for stationary and non-stationary environments. Section III proposes the IBL in detail. Experimental results are discussed in Section IV. We conclude this study in Section V.

II. Related Works

In hashing-based image retrieval problems, all images in the database is hashed to a hash code according to a hashing method. Such that, only the hash codes of images are stored and used to compute similarity for reduction of both space and retrieval time. When a query image arrives, its hash code is computed and compared with hash codes of images in the database. Images with hash codes yielding the smallest Hamming distances from the hash code are retrieved.

In this section, we briefly elaborate on the existing hashing methods for both stationary and non-stationary environments.

Then, the incremental hashing method for non-stationary environments with concept drift is introduced in Section II-C.

A. Current Hashing Methods for Stationary Environments

Current hashing methods can be generally divided into three categories: unsupervised, supervised, and semi-supervised methods. The major difference between these methods is how the semantic information is being used for training hash functions. Unsupervised hashing methods generally train hash functions based on the characteristics of the training data. The Locality sensitive hashing (LSH) [7] is the most representative unsupervised hashing method, which builds hash functions randomly without even considering the distribution the data. Locality-sensitive binary codes from shift-invariant kernels (SKLSH) [8] and Kernelized LSH [9] are variants of the LSH based on kernel methods. The Iterative Quantization [10] finds the rotation matrix of mapped data and hash codes by minimizing a quantization loss function. The spectral hashing (SH) [11] finds efficient hash codes based on the spectral graph partitioning. Based on the SH, the spectral embed hashing [12] is proposed by introducing a new regularizer to the objective function of the SH. The complementary hashing (CH) [13] balances the retrieval precision and the recall effectively by training multiple complementary hash tables. The unsupervised bilinear local hashing [14] learns discriminative binary codes based on local features for image retrieval. The asymmetric cyclical hashing [15] employs short hash codes for stored images to reduce the storage requirement and long hash codes for computing Hamming distance between the query and stored images to improve retrieval accuracies. In contrast to using hyperplanes to partition the data space in majority of hashing methods, the spherical hashing [16] applies hash hyperspheres to produce more efficient hash codes. A bit selection method is proposed in [17], which trains hash functions using different features of data and different hashing methods to build a hash function pool. Then, both the similarity preservation and the angular difference are used to select appropriate hash functions. However, this method is designed to handle stationary environments and ignores the partition balancing problem. Moreover, the optimization process achieving a high overall angular difference ignores the angular difference between individual selected hash function. The projection selection hashing [18] is designed for the SKLSH using the Relief algorithm to improve its bit efficiency. Semantic-assisted visual hashing [19] trains hash functions with the extracted semantic information from auxiliary texts of images. Based on the sparse hashing [20], the sparse hashing with optimized anchor embedding [21] is recently proposed to exploit non-linear projections based on spare representation of data to preserve the complicated geometric structure of data.

When semantic information is available, supervised and semi-supervised hashing methods yield better hashing functions than unsupervised hashing methods′. The LDA hashing [22] is an instance of supervised hashing methods which aims to build similar hash codes for similar image pairs and different hash codes for dissimilar image pairs. The objective function is optimized by the linear discriminant analysis. The Bayesian supervised hashing [23] generates hash codes for training data by iteratively tuning hyperparameters to maximize a posterior estimation while hash codes for queries are generated by a linear regression. The sensitivity based image filtering method [24] is proposed to use radial basis function neural network to filter dissimilar candidate images from multi-hashing. In recent years, we have also witnessed the rapid development of hashing method based on deep neural networks. The supervised semantics-preserving deep hashing [25] employs a latent layer in the deep convolutional neural network as hash functions, and learns hash bits by optimizing an objective function based on both the classification error and other properties of hash codes, i.e. the balancing property. Nonlinear discrete hashing [26] utilizes the non-linear multilayer neural network and learns hash codes based on minimizing the information loss caused by hash projections and maximizing the similarity preservation. Unfortunately, it is impractical to require semantic information for all images in databases for large scale image retrieval problems. Therefore, semi-supervised hashing methods are proposed to use partially labeled databases for hash function training. In general, semi-supervised hashing methods learn hash functions by semantic information in labeled images and partition balancing using unlabeled images. The sequential projection learning hashing (SPLH) [27] trains new hash functions iteratively by correcting errors made by the previous one. A semi-supervised hashing method is proposed based on the SPLH to train each hash function by maximizing the conditional entropy with respect to all previous ones [28]. Another representative semi-supervised hashing method is the Bootstrap SPLH (BSPLH) [29], which trains hash functions to correct errors made by all previous hash functions. Based on the unsupervised method CH and the semi-supervised method SPLH, the semi-supervised dual complementary hashing (DCH) [30] is proposed to learn complementary hash functions and hash tables simultaneously. However, the DCH could not further improve its performance after a small number of hash tables being trained. Thus, a bagging-boosting-based multi-hashing method with query-adaptive re-ranking [31] is proposed recently to handle this problem, which increase the number of samples used for the training of new hash tables and employs a category-specific weighting schema for better retrieval performance.

B. Existing Non-stationary Hashing Methods

Most of current hashing methods are designed for stationary environments only which assume the training database is available in advance and will not change afterward. In contrast, with the rapid increment of digital images in the real world, databases of semantic image retrieval problems are unavoidably and naturally non-stationary and being updated over time. Therefore, online-based hashing methods are proposed to handle this problem. The online kernel-based hashing (OKH) [4] updates hash functions iteratively with a pair of images and corresponding semantic relationship. The adaptive online hashing [32] only updates selected hash functions based on streaming data by the stochastic gradient descent. The online sketching hashing (OSH) [5] updates the

sketch of database dynamically when new images appear. The online supervised hashing [33] updates hash functions in a discriminative manner based on error correcting output codes. In the MIHash [34], the mutual information between the Hamming distance distribution and the similarity relationship of images is used as the objective function for deep neural network to generate hash bits. However, All of these online-based hashing methods, except the unsupervised OSH, require all images in the database and incoming being labeled. This assumption makes them impractical because it is unreasonable to force all images being labeled and manual labeling of incoming images may be infeasible for the huge volume and fast changing Internet environments. The OSH only focuses on the sketch of the whole dataset without any semantic information which is not suitable for distribution changing scenarios of semantic classes. All of the aforementioned methods are not designed for non-stationary environments with concept drifts. Therefore, the incremental hashing (ICH) is proposed recently to deal with semantic image retrieval problems in non-stationary environments with concept drifts. Image retrieval problems on the Internet and for big data are inherently non-stationary and concept drift is expected to appear because of the continuously updating database without central control. However, researches in this area are still very limited. Both the ICH and the proposed IBL in this work serve as one of starters of this research area and provide two different ways to solve this problem.

C. Incremental Hashing for Non-stationary Image Retrieval

In non-stationary environments, every arriving batch of images is usually partly labeled and ratio of images in different semantic classes may not follow the distribution of images in the database. Therefore, the ICH is proposed to deal with both semi-supervised and concept drift properties of semantic image retrieval problems in non-stationary environments.

The ICH trains a hash table using the BSPLH [29] for each arrived image data chunk. Then, the performance of existing hash tables (including the newly trained one and tables in the ICH from last time step) are evaluated by a ranking weight r_i which is computed by the product of $[0, 1]$-normalized γ_i (similarity consistency) and δ_i (code variance) for the i^{th} hash table. If the number of hash tables exceeds a preselected threshold, the ICH removes the hash table yielding the smallest r_i. If Q images are required for a given query, Q images yielding smallest Hamming distances from the query image are returned by each hash table in the ICH to form a candidate set. The rank of each image in the candidate set is computed by the sum of r_i of hash tables returning this image. Finally, Q images yielding the largest ranks are returned as results of retrieval.

By updating the ensemble of hash tables and corresponding ranking weights, the ICH adapts to new non-stationary data environments. The BSPLH prevents repeated hash function (bit) in each hash table of the ICH. However, when the concept drifting is slow, hash functions in hash tables being trained in different time steps may be similar or even the same. Moreover, the ensemble of hash tables requires a large storage cost for large scale retrieval problems. More importantly, the ICH rank the entire hash table with a single weight while different hash functions in a table may yield different significances with respect to a given data environment. These may reduce the performance of the ICH. Therefore, the IBL is proposed to alleviate these weaknesses of the ICH for non-stationary semantic image retrieval problems.

In summary, most of hashing-based image retrieval methods are designed for stationary environments while non-stationary hashing-based image retrieval methods impractically require fully labeled databases. The ICH is able to deal with concept drift for non-stationary image retrievals. However, table-wise semi-supervised incremental hashing in the ICH creates redundant hash bits in different hash tables which may reduce the retrieval efficiency. Therefore, a bit-wise semi-supervised hashing-based image retrieval method for non-stationary environments is eagerly needed. This motivates us to propose the IBL in this work.

III. The IBL

Without losing generality, in non-stationary environment, we assume images arrive in chunks with the same size sequentially for simplicity. The chunk of images arrives at time $t = T$ is denoted as $D_T \in R^{n \times d}$ where n and d denote the number of images in this chunk and the dimensionality of each image, respectively. The IBL works the same when sizes of chunks are different over time. The overview of major processes in the IBL is shown in Figure 1. The third process is the core of the IBL and will be elaborated in detail.

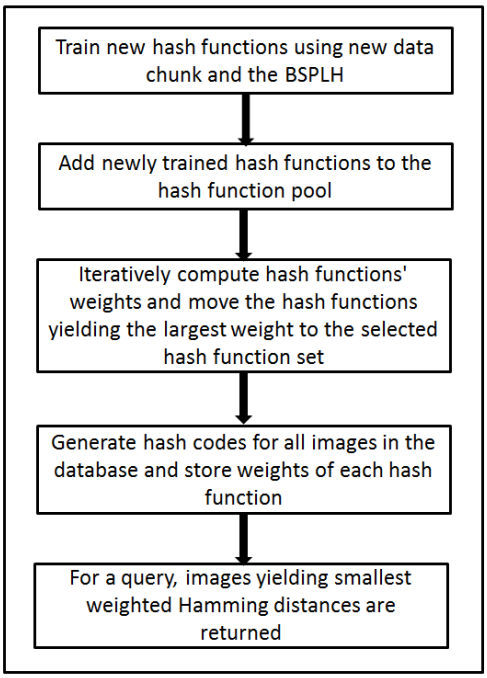


Fig. 1: The overview of major processes in the IBL

At time $t = T$, the IBL learns a new set of b hash functions $W_T = \{h_1, h_2, ..., h_b\}$ based on D_T using the semi-supervised hashing method BSPLH [29] where h_k denotes the k^{th} hash function computed as follows:

$$h_k(x) = sgn(w_k x + a_k) \quad (1)$$

where $sgn(\cdot)$, w_k, and a_k denote the sign function, the projection vector and the intercept vector of the hyperplane of the k^{th} hash function. In case of necessary, the hash function value can be converted to 0-1 binary hash code easily by $\frac{1}{2}(1 + h_k(x))$ [35]. Then, the newly learned hash functions are added to the hash function pool consisting of all hash functions being learned from the beginning till time $T-1$, i.e. $\psi^{(T)} = \{h_1, h_2, ..., h_{Tb}\}$. The two major components of the IBL are hash function selection and hash function weighting by the maximization of a 3-component objective function. In bit selection phase, b hash functions are selected from $\psi^{(T)}$ to form the current set of hash function at time T, i.e. $\Psi^{(T)} = \{h_1, h_2, ..., h_b\}$.

Objective values of each selected hash function are stored in $V = \{v_1, v_2, ..., v_b\}$ to serve as the weight for computing the weighted Hamming distances of images to enhance the retrieval accuracy in the retrieval phase. The weighted Hamming distance between two codes of images x_i and x_j is calculated as follows:

$$H(x_i, x_j) = \sum_{k=1}^{b} \mathrm{v}_k \{\frac{1}{2}(1 + h_k(x_i)) \oplus \frac{1}{2}(1 + h_k(x_j))\} \quad (2)$$

where $H(\cdot)$ and $\oplus$ denote the weighted Hamming distance function and the XOR operator, respectively.

In Section III-A, the bit selection process of the IBL is introduced. The three components of the objective function and the hash function weighting scheme are introduced in Section III-B. Finally, we introduce the light version of the IBL in Section III-C.

A. Hash Function Selection

The subscript time t is ignored for simplicity in this section and weight is calculated at a given time. The objective function value of hash function $h_k \in \psi$ is denoted as υ_k. Three weight components are employed in the objective function to evaluate the performance of hash functions: the information preservation (P), the partition balancing (B), and the bit angular difference (I). These three components evaluate a hash function in three different important aspects, i.e. precision and entropy to the new data, and the dissimilarity between hash functions being selected. The IBL select b hash functions at time T iteratively. The weight component I evaluates the difference between two hash functions, therefore only the P and B are used for the selection of the first hash function in each time T. In this way, the hash function yielding the best information preservation and the best balance partitioning of the space is selected as the first hash bit of the IBL for time t. The bit angular different is not used for the selection of the first bit. In the process to select the first hash function, the objective function value υ_k of $h_k \in \psi$ is calculated as follows:

$$\upsilon_k = P(h_k)B(h_k) \quad (3)$$

The hash function yielding the maximum υ_k is moved from ψ into Ψ as the first selected hash function Ψ_1. Meanwhile, the weight of Ψ_1 is stored as V_1. For the second and following hash functions to the b^{th} hash function selection, the full 3-component objective function is calculated as follows:

$$\upsilon_k = P(h_k)B(h_k)min\{I(h_k, h_i)|h_i \in \Psi\} \quad (4)$$

where $min\{I(h_k, h_i)|h_i \in \Psi\}$ denotes the minimum angular differences between the $h_k \in \psi$ and each $h_i \in \Psi$. For the selection of hash bit, except the first one, the bit angular difference is as important as the information preservation and the partition balancing and used to prevent redundant or highly similar hash bits. The minimum angular different is used instead of the average or the summation because a newly selected hash function should be different from any other selected hash functions. The hash function $h_k \in \psi$ yielding the maximum υ_k is moved from ψ into Ψ and its weight is stored as V_k. The pseudo-code of the IBL is shown in Algorithm 1.

The objective function uses multiplication instead of summation because the multiplication of components relieves the interference of the scale of individual components. The multiplication-based objective function is maximized if and only if all components are maximum. In contrast, the magnitude of summation-based objective function is easily misled by a very large value in an individual component. Although this problem can be relieved by adding scaling parameters in the objective function, its selection for non-stationary environment in every iteration is very time consuming. Therefore, the multiplication is used in our objective functions.

Algorithm 1 IBL Hash Function Selection at time T

Input: the recent chunk of data D_T, the hash functions pool $\psi^{(T-1)}$ before $t = T$.
Output: the selected hash functions set $\Psi^{(T)}$, the weights of selected hash functions $V^{(T)}$.
Initialize: $\Psi^{(T)} = \emptyset, V^{(T)} = \emptyset$

1. Train b hash functions W_T using the BSPLH method and D_T;
2. Add trained b hash functions in W_T into $\psi^{(T-1)}$ to get Tb hash functions set $\psi^{(T)}$, i.e. $\psi^{(T)} = W_T \rightarrow \psi^{(T-1)}$;
3. Calculate υ_k for each $h_k \in \psi^{(T)}$, $k = 1, 2, ..., Tb$ using Eq.(3);
4. Select the hash function $h_i \in \psi^{(T)}$ yielding the largest weight υ_i as the first selected hash function, i.e. $\Psi^{(T)} \leftarrow h_i, V_1^{(T)} = P(h_i)B(h_i)$.
5. Delete the selected h_i from $\psi^{(T)}$;
6. **for** $k = 2$ to b **do**
7. Calculate weights for all hash functions in $\psi^{(T)}$ by Eq.(4);
8. Find the hash function $h_j \in \psi^{(T)}$ yielding the largest weight υ_j;
9. $\Psi^{(T)} \leftarrow h_j, V_k^{(T)} = \upsilon_j$;
10. Deleted the selected h_j from $\psi^{(T)}$.
11. **end for**

B. Weight Components of the IBL Objective Function

The IBL evaluates hash functions based on its performance to the updated data environment in three aspects: information preservation (P), partition balancing (B), and bit angular difference (I). Usually only a small portion of images being added to the database is labeled, hence it is a semi-supervised learning environment. With semantic information provided by labeled images, similar images are expected to share similar hash codes while the differences of hash codes for dissimilar images should be as large as possible. Hence, the information preservation (P) measures the semantic similarity preservation ability of a hash function. On the other hand, each hash function is a hyperplane partitioning the original data space into two parts ($h = -1$ or 1). In information theory mind, a variable with higher entropy contains more information. Therefore, a hash function partitioning the data space evenly to achieve the maximum entropy is preferred [27]. Finally, hash functions selected should be independent with each other to avoid redundancy. Therefore, the minimum bit angular difference between two hash functions should be maximized to prevent similar or redundant hash functions.

1) Information Preservation: For a single hash hyperplane (function) with a good semantic information preservation capability, similar images should be partitioned to the same side of the hyperplane while dissimilar images should be partitioned to different sides. When data distribution changes, a good hash function should still be able to partition similar images to the same side and dissimilar images to different sides of its hyperplane. Therefore, the information preservation of a hash function is computed to evaluate its precision and adaptability to the new data distribution.

Let $X_L \in R^{L\times d}$ be the set of labeled images in the newest data chunk D_T where L denotes number of labeled images. Then, the element S_{ij} for labeled images x_i and x_j in X_L of the pairwise similarity matrix S is calculated as follows:

$$S_{ij} = \begin{cases} +1 & \text{if } (x_i, x_j) \in \varphi_+ \\ -1 & \text{if } (x_i, x_j) \in \varphi_- \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where φ_+ and φ_- denote the set of similar image pairs and the set of dissimilar image pairs, respectively. Let $y_k \in \{+1, -1\}^{L\times 1}$ be the hash function values for images in X_L using the hash function h_k and $y_{ki} \in \{+1, -1\}$ be the hash function value of x_i using h_k. Then, the information preservation (P) for the hash function h_k is calculated as follows:

$$\begin{aligned} P(h_k) &= \sum_{(x_i,x_j)\in\varphi_+} y_{ki}y_{kj} - \sum_{(x_i,x_j)\in\varphi_-} y_{ki}y_{kj} \\ &= \frac{1}{2}(\sum_{i=1}^{L}\sum_{j=1}^{L} S_{ij}(y_{ki}y_{kj}) - L) \\ &= \frac{1}{2}(y_k' S y_k - L) \end{aligned} \tag{6}$$

where y_k' denotes the transpose of y_k.

2) Partition Balancing: Most of images in the database are unlabeled. Without any a priori knowledge, a hash hyperplane evenly partitioning the input space yields the largest entropy and provides the largest amount of information according to the information theory. When new images are added, the data distribution may stew to some direction and therefore a hyperplane evenly partitioning the old database may not be able to evenly partition the updated database. So, the partition balancing capability of a hash function needs to be traced whenever the data changes.

For a hash function h_k, partition balancing (B) evaluates whether the hash hyperplane partition the data space evenly. Let $Y_k \in \{+1, -1\}^{n\times 1}$ denotes the hash function values using hash function h_k for the newest data chunk D_T where n denotes the number of images in this chunk. By regarding one hash function value as a variable, the B of h_k is computed by its entropy as follows:

$$\begin{aligned} B(h_k) = -p(Y_k = 1)log_2(p(Y_k = 1)) - \\ p(Y_k = -1)log_2(p(Y_k = -1)) \end{aligned} \tag{7}$$

where $p(Y_k = 1)$ and $p(Y_k = -1)$ denote the probability of the hash function value equals to 1 and -1 for all images in the data chunk, respectively.

3) Bit Angular Difference: Two same or highly similar hash hyperplanes may both yield high information preservation and partition balancing, but are redundant. Therefore, hash functions being selected by the IBL should yield high scores in the two previous components while being different. Each hash function can be regard as a hyperplane as shown in Eq.(1). Therefore, the redundancy of hash functions can be evaluated by the angular difference between hyperplanes of hash functions. The average angular difference between a hash function and all other hash functions may be misled by a single large angular difference. Therefore, the angular difference between the candidate hash function and the most similar previously selected hash function is used to compute the minimum angular difference. A candidate hash function yielding the maximum minimum angular difference is the most dissimilar to other hash functions and is the most preferable.

A large sinusoidal function value means a angular large difference between two hyperplanes. So, the Bit Angular Difference between two hash functions h_k and h_i is computed by the sinusoidal function value between the two hyperplanes as follows:

$$I(h_k, h_i) = \sqrt{1 - (\frac{w_k w_i}{\parallel w_k \parallel \parallel w_i \parallel})^2} \tag{8}$$

where w_k and $\parallel \cdot \parallel$ denote the projection vector of h_k and the norm function.

C. Light Version of the IBL

With chunks of data arriving continuously, the size of the hash function pool increases without limit. Both the computational time and the storage requirement of the IBL will become intolerable. Therefore, we propose a light version of the IBL (i.e. IBL/L). The IBL/L only stores mb hash functions in the

pool instead of possibly infinitely many hash functions, where m is a non-negative natural number.

At each time step T, a new set of b hash function is learned and added to hash function pool $\psi^{(T)}$. After selecting b hash functions for the current IBL, b hash functions yielding the smallest objective function values are removed from $\psi^{(T)}$ to maintain mb hash functions being stored. Experimental results show that IBL/L using $m = 1$ yields similar performance of the IBL without limiting the value of m.

D. Time and Space Complexity Analysis of the IBL

The IBL trains and selects hash functions adaptively. This method consists of two major steps in iteration: 1) Train a new set of hash functions using the BSPLH; 2) Select the optimal set of hash functions. The time complexity of the whole IBL at time T is $\mathcal{O}((nd^2+bL^2)+(Tb^2L^2)+(Tbn+b^2n)+(Tb^3d^2))$. These four components of the time complexity corresponding to the time complexity of the BSPLH training, the computation of the Information Preservation, the Partition Balancing, and the Bit Angular Difference, respectively.

The space complexity of the IBL is determined by three main parts: hash functions pool, space used to store intermediate variables for computing objective values, and hash codes of images in the database. The space complexities of storing the hash functions pool, storing intermediate variables for computing objective values, and hash codes of images in the database are $\mathcal{O}(Tbd)$, $\mathcal{O}(L + Tb + n + d)$, and $\mathcal{O}(nb)$, respectively. Therefore, the overall space complexity of the IBL at time T is $\mathcal{O}((Tbd) + (L + Tb + n + d) + (nb))$.

When T gets large over time, the number of hash functions in the pool becomes very large (i.e. Tb) which leads to a very large time complexity of the IBL. Therefore the light version is proposed to reduce the time and space complexities of the IBL. Given that m is set to be 1 in this work, the time and space complexities of the IBL/L are $\mathcal{O}(nd^2 + bL^2 + b^2L^2 + Tbn+b^2n+b^3d^2)$ and $\mathcal{O}(bd+L+b+n+d+nb)$, respectively.

IV. Experimental Studies

As far as we know, there is no open database for non-stationary semantic image retrieval problems, therefore, we simulates 15 non-stationary scenarios with different concept drifts in Section IV-A. These 15 scenarios are used to test the IBL and the IBL/L with existing representative hashing methods in Section IV-B.

The IBL is compared with the IBL/L, ICH, the OKH, the OSH, the BSPLH, the SPLH, the SH, and the LSH. The ICH, the OKH, and the OSH are representative hashing methods for non-stationary environments. Both the BSPLH and the SPLH are representative semi-supervised hashing methods for stationary environments. The SH and the LSH are widely used baseline hashing methods. In experiments, parameters of all comparison methods are adjusted according to their corresponding papers or officially released MATLAB codes. Moreover, the IBL without weighting (IBL/-w) is also compared to show the influence of weight to the IBL. The IBL/-w uses the same learning method as of the IBL except it selects hash functions randomly and uses standard Hamming distance for retrieval instead of the weighted Hamming distance in the IBL.

In our experiments, the top 100 precision and the top 1% precision [6] are used to evaluate the performances of different hashing methods. The top 100 precision is computed by the precision of top 100 retrieved images yielding the smallest Hamming distances from the queries for each method. However, in non-stationary environments, the precision of the top 100 retrieved images tends to increase when the number of images in the database increases. In addition to the change of the data distribution, the change of the number of images in database also affects the top 100 precision of hashing methods. Densities of samples in the feature space increases when new chunks of data being added to the database over time. This leads to an overall increment of precision without regarding the performance of hashing methods because more samples are located closely. As we want to test the performance of different hashing methods during the addition of image over time, the top 1% metric is proposed in [6] to adapt the continuous increment of the number of images over time. The top 1% precision evaluate on more retrieved images when the number of images in the database increase. This provides a better evaluation metric for non-stationary relieves the influence of increasing sample density and number of images over time. The number of hash bits is set to be 64 for all hashing methods in experiments. All experiments are repeated 10 times to compute their average results for the final performance assessments.

A. Databases for Non-stationary Environment Simulations

Four real world image databases are used: the CIFAR-10 database [36], the CIFAR-100 database [36], the MNIST database [37], and the NUSWIDE database [38]. The CIFAR-10 database consists of $60,000$ images uniformly distributed in 10 classes e.g. airplane, bird, etc. Each image is a 32×32 pixel color image and is described by a 512-dimensional GIST descriptor. The MNIST database consists of $70,000$ images hand-written digital black-and-white images uniformly distributed in 10 classes i.e. $0, 1, , 9$. Each image is described by 784-dimensional binary features. The CIFAR-100 database consists of $60,000$ 32×32 pixel color images evenly distributed in 100 classes. Moreover, these 100 classes are grouped into 20 super-classes. Each image in the CIFAR-100 database is described by a 512-dimensional GIST feature vector. The NUSWIDE database consists of $269,648$ images belonging to 81 concepts. Each image is described by a 500-dimensional bag of words descriptor based on the SIFT descriptor [38]. In our experiments, $209,347$ images belonging to at least one concept are used and images sharing at least one common concept are regard as similar images.

In this paper, we focus on the two most common concept drift problems: new class appearing and distribution drifting. 15 non-stationary data environments are simulated including 7 new class appearing, 5 distribution drifting, and 3 combined scenarios. Setting of the 15 data environments are shown in Tables II, III, and IV. The CIFAR-10 database, the MNIST database, and the NUSWIDE database are used to simulate

TABLE II: Settings of experiments for new class appearing scenarios

Database	$0 \leq t \leq 5$ ($0 \leq t \leq 10$ for NUSWIDE)	$t > 5$ ($t > 10$ for NUSWIDE)	**Scenario name**
CIFAR-10	Images randomly selected from 5 original classes	Images randomly from 5 original classes and 1 new class	CIFAR10_1
	Same as above	Images randomly from 5 original classes and 3 new classes	CIFAR10_3
	Same as above	Images randomly from 5 original classes and 5 new classes	CIFAR10_5
MNIST	Images randomly selected from 5 original classes	Images randomly from 5 original classes and 1 new class	MNIST_1
	Same as above	Images randomly from 5 original classes and 3 new classes	MNIST_3
	Same as above	Images randomly from 5 original classes and 5 new classes	MNIST_5
NUSWIDE	Images randomly selected from 20 original classes	Images randomly from 20 original classes and 61 new classes	NUSWIDE_20

TABLE III: Settings of experiments for distribution drifting scenarios

Database	$t = 0$	$t > 0$	**Scenario name**
CIFAR-100	Images randomly selected from 20 classes	Each super-class drifts as shown in Figure 3	CIFAR100_20
	Images randomly selected from 15 classes	Same as above	CIFAR100_15
	Images randomly selected from 10 classes	Same as above	CIFAR100_10
	Same as above	Only 6 super-classes drifts as shown in Figure 3	CIFAR100_10_6
	Same as above	Only 3 super-classes drifts as shown in Figure 3	CIFAR100_10_3

TABLE IV: Settings of experiments for combined non-stationary scenarios

Database	$0 \leq t \leq 20$	$21 \leq t \leq 40$	**Scenario name**
CIFAR-100	• 5 stationary super-classes and 5 super-classes drift when $0 \leq t \leq 20$	• 10 new super-classes appear since $t = 21$	CIFAR100_DN
	• 15 original super-classes without any change at $0 \leq t \leq 20$ • 5 new super-classes appear since $t = 6$	• Since $t = 21$, 10 out of 15 original super-classes begin drift as shown in Figure 3 • The 5 new super-classes continuously appear	CIFAR100_ND
	• 5 super classes remain unchanged • 5 super-classes drift at $0 \leq t \leq 20$ • 10 new super-classes appear since $t = 6$	Experiment ends at $t = 20$	CIFAR100_D&N

scenarios with new class appearing, i.e. images from new class(es) will arrive at a specified time. For new class appearing scenarios with the CIFAR10 and the MNIST databases, 5 classes are randomly selected to build data chunks. After time $t > 5$, images from randomly selected new classes (1, 3, and 5 classes in different scenarios) are added into newly appearing data chunks. For the new class appearing scenario with the NUSWIDE database, images belonging to 20 randomly selected concept classes are randomly drawn to form data chunks at the beginning. Due to the complexity of the database at the beginning (20 concepts), new classes appears when $t > 10$ to give all hashing methods more time to learn from the data environment. After 10 time steps, images from all 81 concepts are drawn to form data chunks.

The CIFAR-100 database is used to simulate distribution drifting scenarios. This database consists of 20 super-classes with 5 sub-classes each. In Figure 2, five super-classes in CIFAR-100 database with five sub-classes each are shown. For instance, the super-class *Fish* has sub-classes of *aquarium fish*, *flat fish*, *ray*, *shark*, and *trout*. The distribution drifting scenario for the super-class Fish is simulated by changes over sub-classes as shown in Figure 2. At the beginning, only images of aquarium fish sub-class are available for training. Then, images from other sub-classes of the $Fish$ super-class appear over time. In our experiments, distribution drifting scenarios are simulated by adjusting ratios of images belonging to different sub-classes in the super-class as shown in Figure 3. The ratio of each sub-class appearing in a data chunk follows the Gaussian function. The super-class label is used as the semantic label for images in the same super-class without regarding the sub-class it belonging to.

For distribution drifting scenarios with CIFAR-100 database, we randomly select a number (e.g. 20, 15, and 10) of super-classes firstly and adjust the appearance ratio of sub-classes belonging to each super-class in data chunks over time. Moreover, for the scenarios that 10 super-classes are employed, two additional experiments (i.e. CIFAR100_10_6 and CIFAR100_10_3) are also performed in which distribution drifting happens on randomly selected super-classes only (6 and 3 classes, respectively). Other super-classes do not have distribution drifting and use images from the first sub-class only.

Moreover, 3 combined non-stationary environments are simulated by combining two aforementioned scenarios in either different orders or together to generate more complicated scenarios to validate the performance of the IBL. Setting of these combined scenarios are shown in IV. Both training and testing sets are extracted from the same data environment.

There are 41 data chunks in experiments from $t = 0$ to 40 for both the CIFAR100_ND and the CIFAR100_DN. All other scenarios consists of 21 data chunks from $t = 0$ to 20.

For each time step, a data chunk consisting of a training set and a testing set are built and both sets consists of $1,000$ images each randomly selected according to the characteristic of different non-stationary scenarios. In experiments using the CIFAR-10, the CIFAR-100, and the MNSIT databases, 100 images in the training set are randomly selected to serve as the labeled images while the other 900 images serve as unlabeled images. In the experiment using the NUSWIDE database which images may have multiple labels, 200 images in the training set are randomly selected as the labeled images and the other 800 images serve as unlabeled images. For all experiments, semantic labels of all images in the testing set are used for the performance evaluation. Owing to the fact that the OKH is fully supervised, all $1,000$ training images are provided with labels. However, with this handicap, the OKH still perform worse than both the IBL and the IBL/L.

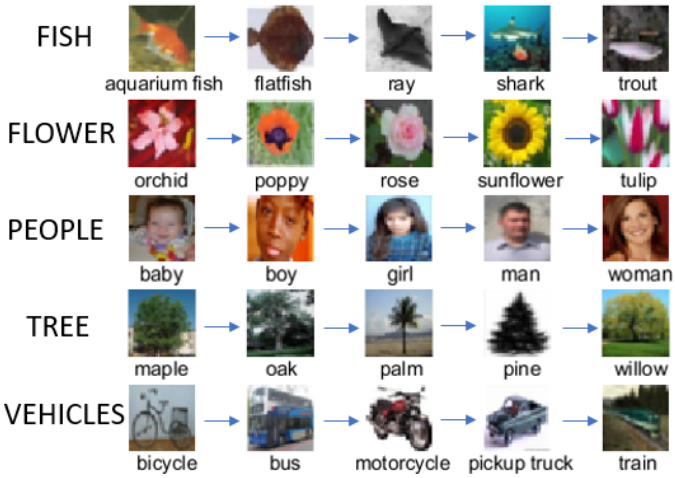


Fig. 2: Distribution drifting for super-classes

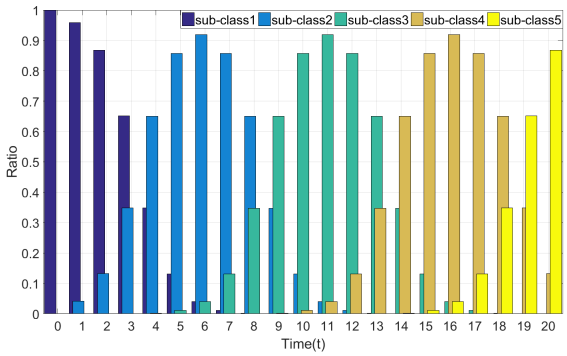


Fig. 3: Appearance ratio of 5 sub-classes in a super-class

B. Comparison with Existing Hashing Methods

Experimental results of new class appearing, distribution drifting, and combined non-stationary scenarios are shown and discussed in Sections IV-B1, IV-B2, and IV-B3, respectively. Their settings are shown in Tables II, III, and IV, respectively.

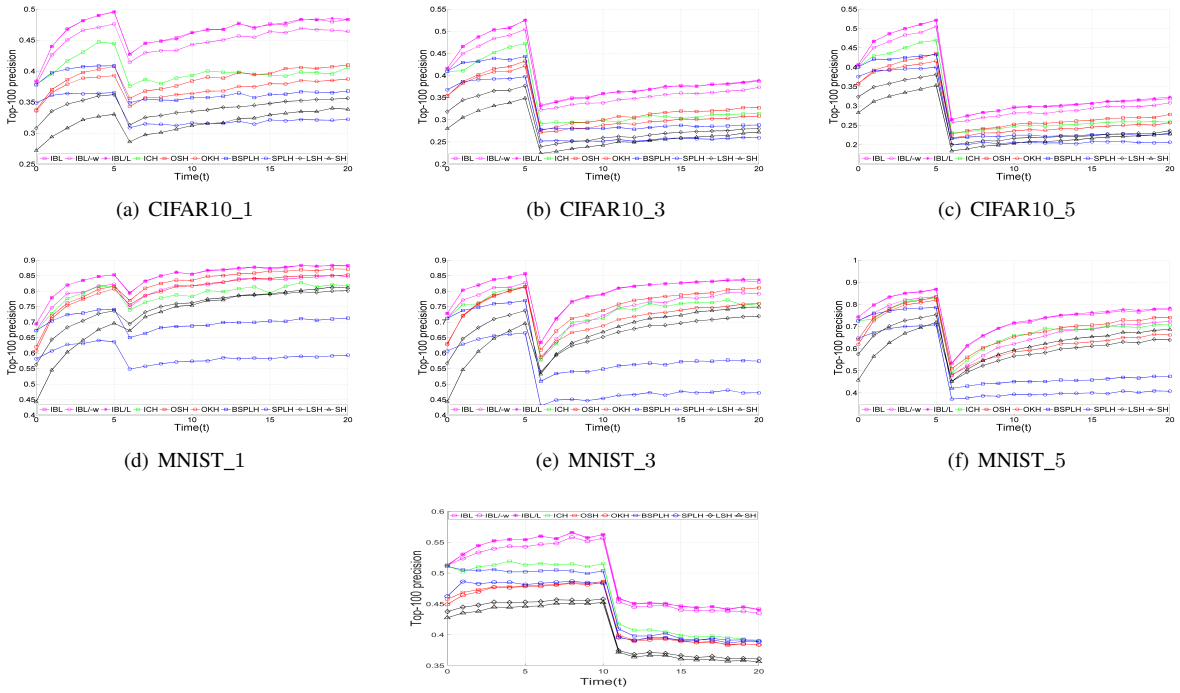
1) Experiments for New Class Appearing: The setting of the 7 scenarios with new class appearing is shown in the Table II. Experimental results of top 100 precision and top 1% precision are shown in Figures 4 and 5, respectively.

According to the Figures 4 and 5, both the IBL and the IBL/L achieve the highest and similar performance. The IBL/-w yields a worse performance in comparison to both the IBL and the IBL/L. This shows the importance of the weighting scheme and hash function selection of the IBL and the IBL/L. The ICH, the OSH, and the OKH yield better performance than hashing methods designed for stationary retrieval environments. These show that existing stationary hashing methods cannot adapt to new non-stationary environments. Moreover, in most scenarios, the performance of stationary semi-supervised hashing methods, i.e. the SPLH and the BSPLH, yield worse performance than unsupervised hashing methods (i.e. the SH and the LSH) after concept drift happening. Stationary semi-supervised hashing methods train hash functions using the original training set given at time $t = 0$ and do not update afterward. Therefore, after concept drifts, the preserved semantic information becomes incorrect and misleads stationary semi-supervised hashing methods. In contrast, stationary unsupervised hashing methods generate hash functions randomly without using the semantic information at $t = 0$. Therefore, when the concept drifts and semantic information provided at $t = 0$ becomes outdated, unsupervised hashing methods suffer less influences in comparison to semi-supervised hashing methods.

The performance of all hashing methods drops more seriously when more new classes appear. Performance drop at the new class(es) appearing is unpreventable because the difficulty of image retrieval increases by the addition of new semantic class(es). According to Figures 4 and 5, both the IBL and its light version IBL/L are more adaptive to new data environments.

2) Experiments with Distribution Drifting: Table III shows the setting of 5 non-stationary scenarios with distribution drifting. Experimental results of top 100 precision and top 1% precision are shown in Figures 6 and 7, respectively.

Figures 6 and 7 show that both the IBL and IBL/L outperform all other hashing methods in experiments. Five different levels of distribution drift complexities with different numbers of super-classes drifting are tested. According to Figures 6 and 7, performances of all hashing methods drop when distribution drifts appear in all semantic classes (the CIFAR100_20, the CIFAR100_15, and the CIFAR100_10). The CIFAR100_10_6 and the CIFAR100_10_3 simulate scenarios that distributions of only 6 and 3 randomly selected super-classes drift out of 10, respectively. Comparing with the other three distribution drifting scenarios, top 1% performances of hashing methods in these two scenarios drop slightly only because distributions of several super-classes are unchanged. When only 3 out of 10 classes change, performances of all hashing methods drop slightly in top 1% precisions and increase in top 100 precisions. As aforementioned, the top 100 precision increases when the number of images increases. Hence, the increments

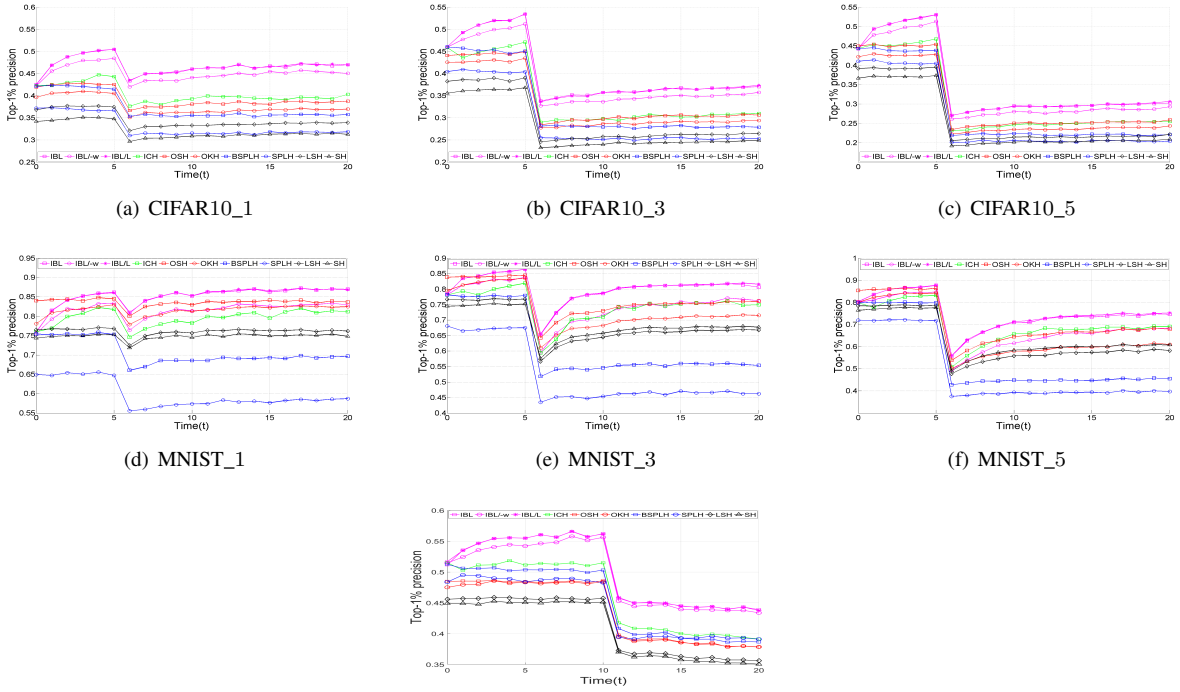


(a) CIFAR10_1 (b) CIFAR10_3 (c) CIFAR10_5

(d) MNIST_1 (e) MNIST_3 (f) MNIST_5

(g) NUSWIDE_20

Fig. 4: The top 100 precisions of 7 non-stationary scenarios with new class appearing.



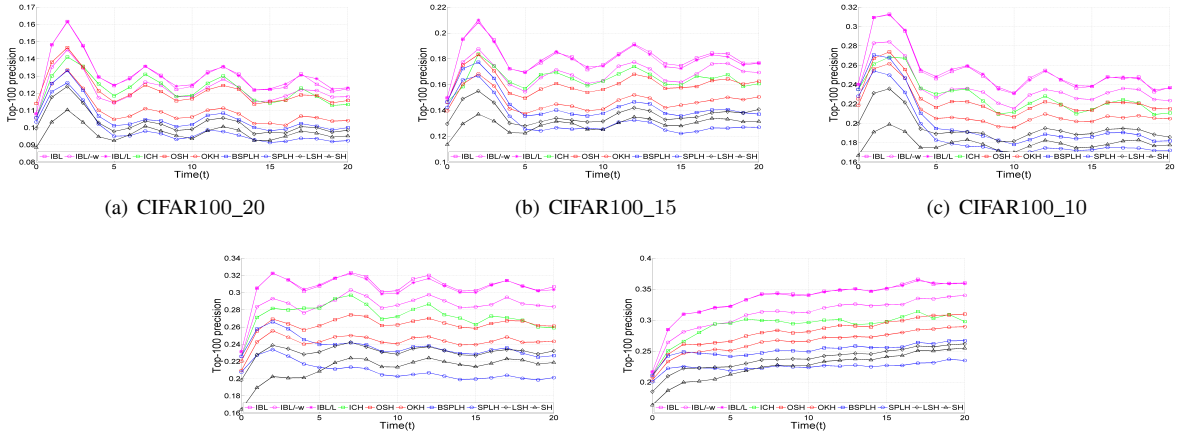
(a) CIFAR10_1 (b) CIFAR10_3 (c) CIFAR10_5

(d) MNIST_1 (e) MNIST_3 (f) MNIST_5

(g) NUSWIDE_20

Fig. 5: The top 1% precision of 7 non-stationary scenarios with new class appearing.

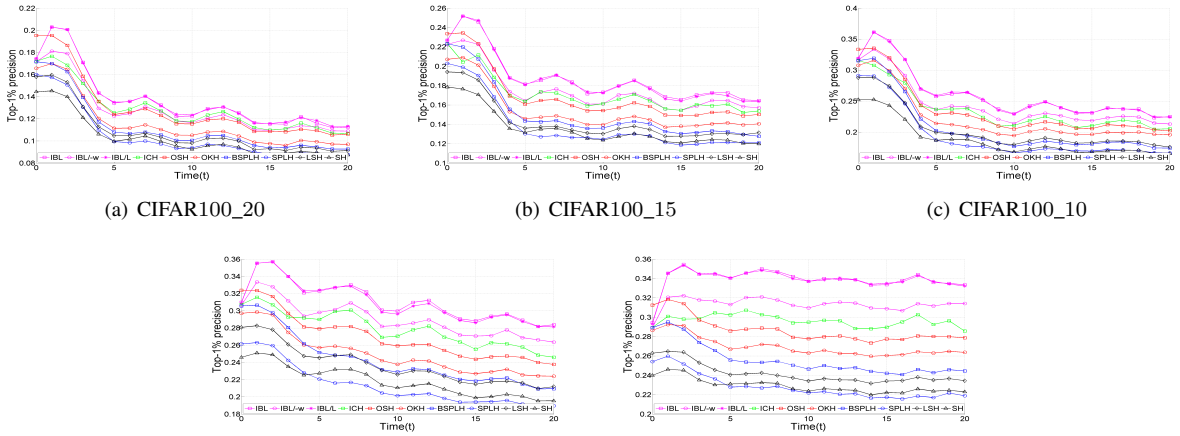
of top 100 precisions in Figure 6(e) may not be solely led by the performance of hashing methods. The performance of all hashing methods increases when the new appearing sub-class occupies a majority ratio in a super-class of the newest data chunk e.g. at $t = 5$, 6, and 7. It is because images in both the training and the testing sets are mostly selected from this sub-class according to the distribution drift shown in Figure 3 and the number of images in this sub-class being added to the database increases significantly at this time period. After this time step, e.g., at $t = 8$ and 9, the distribution



(a) CIFAR100_20 (b) CIFAR100_15 (c) CIFAR100_10

(d) CIFAR100_10_6 (e) CIFAR100_10_3

Fig. 6: The top 100 precision of 5 non-stationary scenarios with distribution drifting.



(a) CIFAR100_20 (b) CIFAR100_15 (c) CIFAR100_10

(d) CIFAR100_10_6 (e) CIFAR100_10_3

Fig. 7: The top 1% precision of 5 non-stationary scenarios with distribution drifting.

of this super-class drifting to another unseen new sub-class gradually and hence the retrieval performance of a super-class decreases. Again, stationary unsupervised hashing methods achieve better performances than stationary semi-supervised hashing methods because they are not misled by the outdated semantic information in $t = 0$.

3) Experiments for Combined Non-stationary Scenarios: Table IV shows the experimental setting of three combined non-stationary scenarios which both new class appearing and distribution drift occur. Top 100 precision and top 1% precision of different scenarios are shown in Figures 8 and 9, respectively.

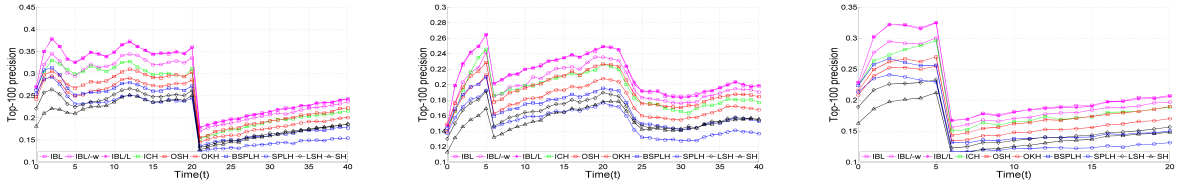
Figures 8 and 9 show that the IBL and the IBL/L outperform other hashing methods in all 3 combined non-stationary scenarios. The CIFAR100_DN scenario and the CIFAR100_ND scenario are simulated with a longer time period, i.e. $t = 0$ to 40, because two types of concept drifts happen consecutively. In contrast, the CIFAR100_D&N scenario is simulated with a period to be the same with other experiments, i.e. time $t = 0$ to 20, because two types of concept drifts happen simultaneously.

For both the CIFAR100_DN and the CIFAR100_ND scenarios, the consecutively occurrence of different types of concept drifts show no obvious interference to others. The fluctuations of the distribution drifting part of the CIFAR100_ND in Figure 9(b), i.e. $t = 21$ to 40 is smaller than that of the CIFAR100_10 in Figure 7(c). It is because distribution drift appears in only 10 out of 20 super-classes in the CIFAR100_ND scenario while distribution drift appears in all 10 super-classes in the CIFAR100_10. Therefore, the complexity of concept drifts (e.g. portions of super-classes drifting) influences the retrieval performances.

For the CIFAR100_D&N scenario, two types of concept drifts happen simultaneously. Performances of all hashing methods drop significantly when 10 new super-classes appear. This is the most complicated scenario in our experiments and all hashing methods yield poor performances. In this most complicated scenario, both the IBL and the IBL/L yield the best performances in comparison to other hashing methods.

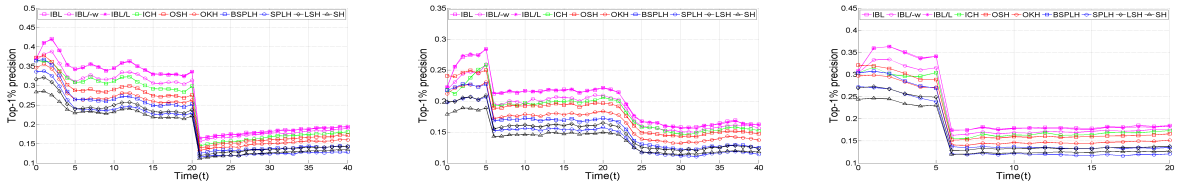
C. Parameter Selection

Generally, a longer hash code yields a higher accuracy but also uses more time and space cost. For the IBL, similar to



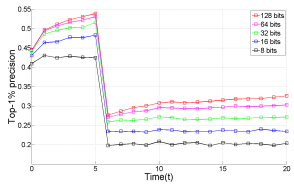
(a) CIFAR100_DN (b) CIFAR100_ND (c) CIFAR100_D&N

Fig. 8: The top 100 precision of 3 combined non-stationary scenarios.



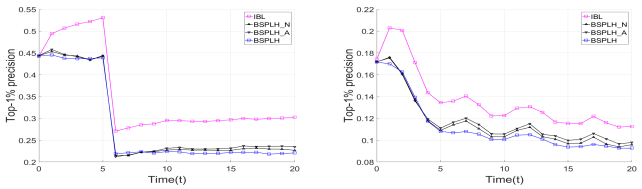
(a) CIFAR100_DN (b) CIFAR100_ND (c) CIFAR100_D&N

Fig. 9: The top 1% precision of 3 combined non-stationary scenarios.



(a) CIFAR10_5 (b) CIFAR100_20

Fig. 10: The top 1% precision of the IBL with different number of hash bits.



(a) CIFAR10_5 (b) CIFAR100_20

Fig. 11: The top 1% precision of the IBL, the BSPLH_N, BSPLH_A, and the BSPLH.

all other hashing methods, the number of hash bits is a key parameter to its performance. A representative scenario from new class appearing (CIFAR10_5) and a representative scenario from distribution drift (CIFAR100_20) are used to show the influence of the number of hash bits to the performance of the IBL.

Figure 10 shows that top 1% precisions improve when the number of hash bits increase in both scenarios. However, the doubling the number of hash bits does not yield a double in the performance. The improvement between 128 bits and 64 bits is smaller than that between 64 bits and 32 bits. Therefore, by the trade-off of computational time and overall performances, 64 bits are used for all hashing methods in our experiments.

D. Comparison with Modified Stationary Hashing Methods

The IBL, the ICH, the OKH, and the OSH are all dynamic hashing methods which update hash functions whenever a new data chunk arrives. The experiments in previous sections show that existing stationary hashing methods are not suitable for non-stationary environments. A direct idea to apply these methods to non-stationary environments is to re-train their hash functions over time using the newest data chunk or all stored data. Therefore, we perform the comparison between the IBL and existing stationary hashing method with aforementioned simple modifications to show the superiority of the IBL. Among those stationary hashing methods in comparison, the BSPLH is the most state-of-the-art semi-supervised hashing method. Moreover, the IBL employs the BSPLH to train new hash functions when new data chunk arrives. Therefore, two modified versions of the BSPLH (i.e. BSPLH_N and BSPLH_A) for non-stationary environments are performed for comparison. When a new data chunk arrives, the BSPLH_N updates hash functions by using the training set in the newest data chunk while the BSPLH_A updates hash functions using all stored data as the training set. Only labeled images in the training set of the newest data chunk are used as supervise information for both methods because they reflect the current data distribution of semantic classes. The IBL is compared with the original BSPLH, the BSPLH_N, and the BSPLH_A on two representative scenarios i.e. CIFAR10_5 and CIFAR100_20. Experimental results are shown in Figure 11.

The original stationary BSPLH is used as the baseline method and shown in Figure 11. The BSPLH yields the worst result because its hash functions are trained based on the initial training set only which is very different from the current data environment. Both the BSPLH_N and the BSPLH_A update hash functions over time utilizing information from the newest data chunk, therefore they yield a bit better retrieval performance than the original stationary BSPLH but much

TABLE V: Average top 1% precision of hashing methods in 15 non-stationary data environments

Scenario name	LSH	SH	SPLH	BSPLH	OSH	OKH	ICH	IBL
CIFAR10_1	0.346±0.021∗	0.320±0.019∗	0.331±0.047∗	0.374±0.041∗	0.394±0.028∗	0.376±0.028∗	0.404±0.034∗	**0.467±0.038**
CIFAR10_3	0.295±0.019∗	0.277±0.012∗	0.296±0.034∗	0.329±0.023∗	0.341±0.017∗	0.328±0.019∗	0.345±0.018∗	**0.401±0.017**
CIFAR10_5	0.265±0.011∗	0.250±0.008∗	0.262±0.024∗	0.284±0.017∗	0.307±0.009∗	0.289±0.011∗	0.307±0.011∗	**0.353±0.009**
MNIST_1	0.761±0.046∗	0.748±0.043∗	0.597±0.088∗	0.707±0.097∗	0.836±0.038	0.815±0.042#	0.796±0.042∗	**0.851±0.031**
MNIST_3	0.683±0.025∗	0.686±0.021∗	0.520±0.056∗	0.615±0.059∗	0.766±0.018∗	0.729±0.026∗	0.744±0.029∗	**0.803±0.023**
MNIST_5	0.623±0.022∗	0.637±0.013∗	0.485±0.046∗	0.548±0.036∗	0.707±0.010∗	0.653±0.023∗	0.698±0.011∗	**0.748±0.009**
NUSWIDE_20	0.412±0.005∗	0.407±0.007∗	0.443±0.014∗	0.452±0.004∗	0.438±0.002∗	0.436±0.003∗	0.460±0.004∗	**0.501±0.004**
CIFAR100_20	0.109±0.007∗	0.103±0.003∗	0.105±0.005∗	0.112±0.004∗	0.129±0.005∗	0.116±0.006∗	0.129±0.003∗	**0.137±0.004**
CIFAR100_15	0.143±0.007∗	0.135±0.006∗	0.138±0.013∗	0.151±0.012∗	0.169±0.009∗	0.154±0.010∗	0.171±0.008∗	**0.188±0.009**
CIFAR100_10	0.204±0.015∗	0.188±0.008∗	0.194±0.019∗	0.208±0.018∗	0.235±0.014∗	0.221±0.014∗	0.235±0.012∗	**0.261±0.011**
CIFAR100_10_6	0.237±0.019∗	0.218±0.014∗	0.213±0.020∗	0.243±0.020∗	0.270±0.016∗	0.250±0.018∗	0.279±0.017∗	**0.311±0.017**
CIFAR100_10_3	0.242±0.020∗	0.229±0.016∗	0.228±0.017∗	0.256±0.023∗	0.287±0.021∗	0.270±0.022∗	0.296±0.020∗	**0.339±0.021**
CIFAR100_DN	0.196±0.016∗	0.183±0.012∗	0.19±0.0210∗	0.209±0.026∗	0.231±0.018∗	0.216±0.022∗	0.242±0.020∗	**0.269±0.019**
CIFAR100_ND	0.151±0.023∗	0.140±0.017∗	0.145±0.019∗	0.159±0.023∗	0.182±0.025	0.168±0.024∗	0.184±0.025	**0.201±0.025**
CIFAR100_D&N	0.170±0.021∗	0.157±0.014∗	0.159±0.021∗	0.179±0.026∗	0.201±0.021$	0.186±0.021∗	0.204±0.022#	**0.227±0.023**

worse than the IBL. However, both the BSPLH_N and the BSPLH_A ignore the data distribution information of existing learned data while these learned data with old data distribution is also used for retrieval. This shows that simply re-training hash functions using newly appearing data is not enough for adapting non-stationary image retrieval problems. More accurate bit weighting and selection schemes are necessary for the adaptation to new non-stationary environments. This is also the main motivation of the proposal of the IBL.

E. Statistical Significance of Experimental Results

In this section, we apply t-test to the average top 1% precision of all scenarios to show the statistical significance of the IBL performance over all other existing hashing methods in our experiments. The average top 1% is computed by averaging top 1% precision values over all time steps and the average top 1% precisions of 15 scenarios of hashing methods are shown in Table V. The t-test is performed between the IBL and other methods for each scenario. Symbols $*$, \$, $\#$ in Table V denote that the IBL outperforms the corresponding hashing method with 99%, 98%, and 95% statistical significance, respectively.

According to Table V, the IBL achieves the highest average top 1% precisions in all non-stationary scenarios. More importantly, the IBL outperforms other hashing methods with 99% significance in 94.3% (99 out of 105) cases. For the 7 new class appearing scenarios, the IBL outperforms other hashing methods with 99% significance in 47 out of 49 (95.9%) cases. For the 5 distribution drifting scenarios, the IBL outperforms other hashing methods with 99% significance in all 35 (100%) cases. For the 3 combined non-stationary scenarios, the IBL outperforms other hashing methods with 99% significance in 17 out of 21 (81.0%) cases.

Therefore, we conclude that the IBL outperforms other hashing methods for semantic image retrieval problems in non-stationary environments with statistical significance.

V. Conclusions

Semantic image retrieval problems in big data and Internet environments are inherently non-stationary and with concept drifts. However, most of current hashing methods are designed for stationary environments. The very few online hashing methods are proposed to solve the image retrieval problems in non-stationary data environment without regarding concept drift issues and impractically requiring fully labeled image databases. The semi-supervised ICH may generate duplicate or highly similar hash bits in different hash tables learned in different time steps. Therefore, the IBL is proposed in this work to improve the hash efficiency and adaptability to concept drifts in non-stationary environments by selecting and weighting hash bits individually by using a 3-component weight. Experimental results show that the IBL outperforms other stationary and non-stationary hashing methods.

The IBL, as one of the very limited works on hashing for semantic image retrieval in non-stationary environments with concept drift, serves as a milestone and starter of this important research area. However, there are still many future works to be done to further explore this area. The major drawback of the IBL is that new hash functions being learned in each time step. Updating hash bits per time step may be unnecessary when concept does not drift in some time steps. Therefore, one of our future works is to develop a concept drift detection method specifically designed for semantic image retrieval problems. With the concept drift detection method, one may design strengths or types of concept drifts to trigger the updating of new hash bits or tables. It is unknown that whether current concept drift detection techniques developed for pattern classification problems to be suitable for semantic image retrieval or not. Nonetheless, they will serve as a good starting point to research on the concept drift detection for semantic image retrieval problems in non-stationary environments.

Furthermore, another drawback of the IBL is the pre-selected number of hash bits being learned in each time

step. This may not be feasible to pre-determine the degree of concept drifts in the non-stationary environment before data chunks arrival. Therefore, the current version of IBL with pre-selected number of hash bits being learned per time step is primitive and may be inefficient. A changeable number of hash bits being learned in each time step may be more efficient for a given strengths and types of concept drifts in a given time step. Meanwhile, a new optimization method of hash bits or tables will also be needed to accommodate strengths and types of concept drift and also both semantic and unlabeled distribution information provided by data chunk per time step.

In addition to the non-stationary environment, imbalanced class distribution and multi-label images provide further challenges to the IBL. An extended version of the IBL needs to be researched to adapt to those highly complicated data environment.

Last but not the least, if we detect the type and the strength of a concept drift in a time step, can we reverse the drifted data back to the original distribution? By this, off-the-shelf hashing methods for stationary environments may provide good retrieval performances.

ACKNOWLEDGMENT

This work was supported by the National Natural Science Foundation of China under Grants 61272201 and 61572201, the Fundamental Research Funds for the Central Universities (2015ZZ023 and 2017ZD052), and China Scholarship Council (201706150058).

REFERENCES

[1] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *Proc. IEEE Conf. Comput. Vis. Patt. Recognit.* IEEE, 2008, pp. 1–8.

[2] A. Beygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 97–104.

[3] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Trans. Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[4] L.-K. Huang, Q. Yang, and W.-S. Zheng, "Online hashing," in *Proc. of International Joint Conference on Artificial Intelligence*, 2013, pp. 1422–1428.

[5] C. Leng, J. Wu, J. Cheng, X. Bai, and H. Lu, "Online sketching hashing," in *Proc. IEEE Conf. Comput. Vis. Patt. Recognit.*, 2015, pp. 2503–2511.

[6] W. W. Ng, X. Tian, Y. Lv, D. S. Yeung, and W. Pedrycz, "Incremental hashing for semantic image retrieval in nonstationary environments," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3814–3826, 2017.

[7] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. of the Twentieth Annual Symposium on Computational Geometry*, 2004, pp. 253–262.

[8] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. of Conf. and Workshop on Neural Information Processing Systems*, 2009, pp. 1509–1517.

[9] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. of IEEE Int. Conf. on Computer Vision*, 2009, pp. 2130–2137.

[10] 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. on Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, 2013.

[11] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. of Advances in Neural Information Processing Systems*, 2009, pp. 1753–1760.

[12] L. Chen, D. Xu, I. W.-H. Tsang, and X. Li, "Spectral embedded hashing for scalable image retrieval," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1180–1190, 2014.

[13] H. Xu, J. Wang, Z. Li, G. Zeng, S. Li, and N. Yu, "Complementary hashing for approximate nearest neighbor search," in *Proc. of IEEE Int. Conf. on Computer Vision*. IEEE, 2011, pp. 1631–1638.

[14] L. Liu, M. Yu, and L. Shao, "Unsupervised local feature hashing for image similarity search," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2548–2558, 2016.

[15] Y. Lv, W. W. Ng, Z. Zeng, D. S. Yeung, and P. P. Chan, "Asymmetric cyclical hashing for large scale image retrieval," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1225–1235, 2015.

[16] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing: binary code embedding with hyperspheres," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2304–2316, 2015.

[17] X. Liu, J. He, B. Lang, and S.-F. Chang, "Hash bit selection: a unified solution for selection problems in hashing," in *Proc. IEEE Conf. Comput. Vis. Patt. Recognit.*, 2013, pp. 1570–1577.

[18] Z. Zeng, Z. Wu, and W. W. Ng, "Projection selection hashing," in *Proc. IEEE Conf. Wavelet Analysis and Pattern Recognition*. IEEE, 2015, pp. 185–191.

[19] L. Zhu, J. Shen, L. Xie, and Z. Cheng, "Unsupervised visual hashing with semantic assistant for content-based image retrieval," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 2, pp. 472–486, 2017.

[20] F. Shen, C. Shen, Q. Shi, A. van den Hengel, Z. Tang, and H. T. Shen, "Hashing on nonlinear manifolds," *IEEE Trans. on Image Processing*, vol. 24, no. 6, pp. 1839–1851, 2015.

[21] Y. Guo, G. Ding, L. Liu, J. Han, and L. Shao, "Learning to hash with optimized anchor embedding for scalable retrieval," *IEEE Trans. on Image Processing*, vol. 26, no. 3, pp. 1344–1354, 2017.

[22] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua, "Ldahash: Improved matching with smaller descriptors," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 34, no. 1, pp. 66–78, 2012.

[23] Z. Hu, J. Chen, H. Lu, and T. Zhang, "Bayesian supervised hashing," in *Proc. IEEE Conf. Comput. Vis. Patt. Recognit.*, 2017, pp. 6348–6355.

[24] W. W. Ng, J. Li, S. Feng, D. S. Yeung, and P. P. Chan, "Sensitivity based image filtering for multi-hashing in large scale image retrieval problems," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 5, pp. 777–794, 2015.

[25] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hash via deep convolutional neural networks," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 437–451, 2018.

[26] Z. Chen, J. Lu, J. Feng, and J. Zhou, "Nonlinear discrete hashing," *IEEE Trans. Multimedia*, vol. 19, no. 1, pp. 123–135, 2017.

[27] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, 2012.

[28] W. W. Ng, Y. Lv, Z. Zeng, D. S. Yeung, and P. P. Chan, "Sequential conditional entropy maximization semi-supervised hashing for semantic image retrieval," *International Journal of Machine Learning and Cybernetics*, vol. 8, no. 2, pp. 571–586, 2017.

[29] C. Wu, J. Zhu, D. Cai, C. Chen, and J. Bu, "Semi-supervised nonlinear hashing using bootstrap sequential projection learning," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1380–1393, 2013.

[30] P. Li, J. Cheng, and H. Lu, "Hashing with dual complementary projection learning for fast image retrieval," *Neurocomputing*, vol. 120, pp. 83–89, 2013.

[31] W. W. Ng, X. Zhou, X. Tian, X. Wang, and D. S. Yeung, "Bagging–boosting-based semi-supervised multi-hashing with query-adaptive re-ranking," *Neurocomputing*, vol. 275, pp. 916–923, 2018.

[32] F. Cakir and S. Sclaroff, "Adaptive hashing for fast similarity search," in *Proc. of IEEE Int. Conf. on Computer Vision*, 2015, pp. 1044–1052.

[33] ——, "Online supervised hashing," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 2606–2610.

[34] F. Cakir, K. He, S. A. Bargal, and S. Sclaroff, "Mihash: Online hashing with mutual information," in *Proc. of IEEE Int. Conf. on Computer Vision*, 2017, pp. 437–445.

[35] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, "Learning to hash for indexing big datała survey," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57, 2016.

[36] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[37] J. Song, Y. Yang, X. Li, Z. Huang, and Y. Yang, "Robust hashing with local models for approximate similarity search," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1225–1236, 2014.

[38] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "Nus-wide: a real-world web image database from national university of singapore," in *Proceedings of the ACM international conference on image and video retrieval*. ACM, 2009, p. 48.