Concept Preserving Hashing for Semantic Image Retrieval with Concept Drift

Xing Tian, Wing W. Y. Ng*, Hui Wang
*corresponding author

Abstract—**Current hashing-based image retrieval methods mostly assume that the database of images is static. However, this assumption is not true in cases where the databases are constantly updated (e.g. on Internet) and there exists the problem of concept drift. Online (a.k.a. incremental) hashing methods are proposed recently for image retrieval where the database is not static. However, they have not considered the concept drift problem. Moreover, they update hash functions dynamically by generating new hash codes for all accumulated data over time which is clearly uneconomical. In order to solve these two problems, *Concept Preserving Hashing* (CPH), is proposed. In contrast to existing methods, CPH preserves the original concept, i.e., the set of hash codes representing a concept is preserved over time, by learning a new set of hash functions to yield the same set of hash codes for images (old and new) of a concept. The objective function of CPH learning consists of three components: isomorphic similarity, hash codes partition balancing, and heterogeneous similarity fitness. Experimental results on 11 concept drift scenarios show that CPH yields better retrieval precisions than existing methods and does not need to update hash codes of previously stored images.**

Index Terms—**Image Retrieval, Hashing, Non-stationary Environment, Concept Drift, Concept Preserving.**

I. Introduction

With the rapid development of the Internet and the wide usage of visual devices, multimedia data grows explosively. There is a huge volume of images on the Internet and this volume is getting bigger every day. It has been reported that about 350 million photos are uploaded to Facebook every day [1]. Efficient retrieval of relevant images from a large and non-stationary database of images has become a challenging problem. To tackle this challenge, not only should the huge volume of the database be taken into consideration, but also the very dynamic nature of the database. As the database is dynamically updated, the (data) distribution of the classes in the database, i.e. *semantic concepts*, may change – a phenomenon called *concept drift* [2]. New concepts may also emerge as new images are added to the database. For examples, as a new and unprecedented product is released or a new actor is getting popular, a lot of images of these new objects (product or actor) are added to the database thus resulting in the emergence of new concepts. Moreover, as new images are added, the distribution of existing concepts may change. For example, an upgrade of BMW 520 is released or the character "Kungfu Panda" is created, resulting in changes of the distribution of classes (BMW 520 and "Panda"). These changes are unavoidable in non-stationary environments and will undoubtedly increase the difficulty of image retrieval.

Content-Based Image Retrieval (CBIR) is an approach to image retrieval, which has been extensively researched in recent years. It seeks to retrieve similar images for a query image based on the content of images, rather than the meta data of images. Tree-based methods, e.g. the k-d tree [3] and the cover tree [4], are a class of methods for content-based image retrieval. They build tree structures based on features of the image data. However, tree-based methods generally suffer from the curse of dimensionality problem [5]. Moreover, the storage cost of tree structure is prohibitively high for real world applications.

Hashing methods are another class of methods for CBIR. They typically have sub-linear complexity and low space cost, thus have become mainstream methods for CBIR. Hashing methods generate hash codes to project images from a high dimensional feature space onto a low dimensional binary Hamming space. In the original feature space, hash functions (hyperplanes) partition the feature space into many hash buckets. For a hash hyperplane, the two sides of this hyperplane correspond to two binary hash values of 0 and 1. Therefore, K hash hyperplanes generate a K-bit hash code for each image in the database according to the relative location of this image with respect to different hash hyperplanes. In fact, the hash code of an image is the ID of the hash bucket that the image is in, and every hash bucket has a unique hash code. Finally, Hamming distance is calculated for two hash codes to evaluate the similarity between their respective images. An ideal hashing method is one such that similar images should have hash codes that have zero or very small Hamming distance while dissimilar images should have hash codes that have large Hamming distance. Many hashing methods have been proposed to provide different ways of generating hash functions, such as Local Sensitive Hashing (LSH) [6], the multi-hashing method Dual Complementary Hashing (DCH) [7], and the semi-supervised method BSPLH [8].

Most of the existing hashing methods assume that the databases are static, however, many real-world image databases are in fact non-stationary. Moreover, the distribution of

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

Hui Wang (h.wang@ulster.ac.uk) is with the School of Computing, Ulster University, Jordanstown, United Kingdom, BT370QB.

This work was supported by National Natural Science Foundation of China under Grants 61572201 and 61876066, Guangzhou Science and Technology Plan Project 201804010245, EU Horizon 2020 Programme (700381, ASGARD), and China Scholarship Council (201706150058).

existing concepts may change over time. When these changes occur, the performance of pre-trained hash functions drops accordingly. Recently, several non-stationary hashing methods have been proposed to deal with non-stationary image retrieval problems, including OKH [9], OSH [10], OH [11] and ICH [12]. However, all of these methods, except ICH, assume that data appear in an online manner and ignore the concept drift problem. Moreover, they all need to update the hash codes for all images in the database using the newest hash functions. This is clearly inefficient and a waste of resources. In order to solve these two problems, we propose a new hashing method, *Concept Preserving Hashing*, which projects images of the same concept from different time steps with different concept drifts to the same set of hash codes. Therefore, images in the database only need to be hashed once when they are added to the database and no update of hash code is needed. In this way, data distribution of semantic classes is preserved in the hash projected space by hash codes for non-stationary environments with concept drifts. We also devise a fast computation method for generating new hash functions in every time step for CPH.

The main contributions of this paper are as follows:

- CPH is the first supervised hashing method for image retrieval in non-stationary environment with concept drift.
- CPH learns hash functions based on new images and old reference images to generate hash codes for new images only. Images of the same concept are mapped to its fixed area in Hash projected space. To our best knowledge, CPH is the first incremental hashing method which generates hash codes for new images only without updating hash codes of old images.
- A fast training method of hash functions is proposed in this paper. The objective function of CPH consists of isomorphic similarity, hash code partition balancing, and heterogeneous similarity fitness. The optimal hash functions for the newest data chunk can be computed directly, which is practical for online retrieval task.
- Experimental results show that CPH outperforms all other comparable stationary and non-stationary methods, including the multi-hashing method ICH, in all 11 non-stationary data scenarios.

The paper is organized as follows. In Section II, related hashing methods are reviewed, especially the existing non-stationary hashing methods. CPH is presented in Section III. Experimental results of CPH and other comparable methods are shown and discussed in Section IV. We conclude the paper in Section V.

II. Related Works

Hashing methods have been widely used for large scale image retrieval problems. Most of existing hashing methods assume the database of images is static. In recent years, several online hashing methods are proposed to update hash functions for image retrieval in non-stationary environments. Representative methods for both stationary and non-stationary hashing are introduced in Section II-A and II-B, respectively. More detailed information about existing hashing methods can be founded in [13].

A. Existing Stationary Hashing Methods

According to whether semantic information is utilized for training, current hashing methods can be generally divided into three types, i.e. unsupervised, supervised, and semi-supervised hashing methods, respectively.

Locality Sensitive Hashing (LSH) [6] is one of the most popular unsupervised hashing methods, which generates hash functions randomly. Several variants of LSH are proposed in [14] [15] [16]. Iterative Quantization (ITQ) [17] learns a rotation matrix of projected images iteratively to minimize the quantization loss. The MLSH-ITQ method [18] learns hash projections with preserving Euclidean distances between images and employs ITQ to minimize the loss of thresholding. Complementary Hashing [19] employs multiple hash tables in which the images already hashed correctly by previous hash tables will not be used again for training the next hash tables. Asymmetric Cyclical Hashing [20] generates long hash codes for query and short hash codes for stored images to achieve both high accuracy and low storage cost. Spectral Hashing [21] generates hash codes based on graph partitioning. Spherical Hashing [22] [23] partitions the image database using hyperspheres instead of hyperplanes to generate hash codes. The bit selection hashing methods are proposed in [24] [25] to select optimal combinations of hash functions based on both similarity preservation capabilities of each hash function and independence between hash functions. Based on sparse hashing [26], the nonlinear Sparse Hashing with Optimized Anchor Embedding [27] selects the optimal anchor set utilized in sparse representation. Distributed Graph Hashing [28] trains hash functions based on data distributed on multiple agents. Kernel Reconstructive Hashing is proposed in [29] which generates hash codes based on the local distribution information of data. Adaptive Binary Quantization [30] learns discriminative hash functions for discovered prototypes and assigns unique hash code to each prototype. Ordinal Constraint Hashing [31] generates hash codes based on the permutation relation information of data instead of pairwise similarity.

Supervised hashing methods train hash functions using semantic information and generally achieve better retrieval performances. The LSH with learned metric [32] employs a learned Mahalanobis metric to preserve the semantic similarity information of images. The LDA hash [33] generates the objective function by minimizing the difference of projected samples between similar image pairs and maximizing the difference for dissimilar image pairs based on Linear Discriminant Analysis. The spectral hashing with semantically consistent graph [34] utilizes the pairwise similarity information between images to optimize the graph Laplacian directly, rather than the Euclidean distance in the original spectral hashing method. Nonlinear Discrete Hashing [35] employs a multi-layer neural network to preserve the local structure of images and learn hash codes. Supervised Discrete Hashing [36] generates optimal hash codes for all images by minimizing the loss function of classifier. The sensitivity-based image filtering method (SIF) is proposed in [37], which firstly utilizes two descriptors to score the hash buckets and candidate images. Then, based on the trained radial basis function neural network, the dissimilar candidate images from multiple hash tables are filtered. Error correcting input and output coding proposed in [38] learns hash codes based on distribution preservation and error correction. In recent years, various deep-neural-network-based hashing methods have also been proposed. For instance, deep supervised hashing (DSH) [39] is a representative deep hashing method which generates discriminative hash codes based on CNN and pairwise similarities of images. Another deep hashing method is proposed in [40] to handle the multi-label images retrieval. The hash code is divided into multiple pieces and each piece belongs to a different category. Triplet-based deep hashing [41] trains hash functions based on the relative relationships among three samples to solve cross-modal retrieval problem.

By combining the advantages of both unsupervised and supervised hashing methods, the semi-supervised hashing methods are proposed which utilize the data distribution information and semantic similarity information simultaneously. The semi-supervised Multi-graph Hashing [42] employs multiple modalities of data to form a weighted multi-graph as a more comprehensive description of dataset. Sequential Projections Learning Hashing (SPLH) [5] learns each hash function by corrects the errors made by the previous one. BSPLH [8] also learns hash functions one by one but treats all previously learned hash functions holistically. Each hash function is trained by correcting the errors made by all previous ones. A semi-supervised hashing (SCEM-SSH) is proposed in [43], which trains hash functions by maximizing the joint entropy of all hash functions. Dual Complementary Hashing (DCH) [7] employs the SPLH to train multiple hash tables sequentially in which the pairwise label matrix is updated iteratively to strengthen the training of next hash table. Based on DCH, BBSHR [44] is proposed which increases the number of pairwise similarities of images to train the multiple hash tables.

B. Existing Non-stationary Hashing Methods

The data environments in the real world are mostly non-stationary, while most of existing hashing methods retrieve relevant images from a given static database. Concerning this issue, several non-stationary hashing methods are proposed in recent years, including Online Kernel-based Hashing (OKH) [9], Online Sketching Hashing (OSH) [10], Online Hashing (OH) [11] and Incremental Hashing (ICH) [12]. OKH and OH both update hash functions based on pairs of images with their similarity in an online manner. OSH updates the sketch of the database dynamically and trains new hash functions based on the structural information of the new sketch. Online Supervised Hashing [45] learns and adapts hash functions in a discriminative manner based on error correcting output codes. The adaptive hashing [46] updates hash functions iteratively based on streaming data. MIHash [47] utilizes mutual information between distributions of Hamming distance of similar and dissimilar images to evaluate the performance of hash table. Then optimal hash codes are learned by optimizing the mutual information objective.

However, all the aforementioned online hashing methods only assume the data appearing in an online manner without considering the concept drift problem. In existing non-stationary hashing methods, ICH [12] is the only hashing method designed for image retrieval with concept drift. ICH employs multiple hash tables to record the semantic information over time in a distributed manner. Hash tables generated by ICH are weighted according to their performance to the current data environment. ICH achieves promising retrieval precision in non-stationary environments with concept drift.

Unfortunately, training a new hash table using BSPLH in ICH at each time step is time consuming and inefficient for practical use. Moreover, the storage cost and retrieval time of ICH is high because each image is represented by multiple hash codes. Last but not least, the major drawback of all existing non-stationary hashing methods is the re-computation of hash codes for all images in the database whenever new hash functions are trained. This is very time consuming and may be prohibitive for solving large scale image retrieval problems. Therefore, in this work, a novel method, CPH, is proposed to deal with non-stationary image retrieval problems with concept drift which trains new hash functions quickly and generate unique single hash code for each image. For CPH, the hash codes of previous images will not be updated when new set of hash functions are learned. As far as we know, CPH is the only work to keep static hash codes for all data when new data appear over time, which is meaningful and significant.

III. Concept Preserving Hashing

Instead of adapting hash functions to concept drifts, CPH preserves the distribution of the original concept in the hash projected space and projects newly drifted concept back to the same area of original concept. This is similar to human mind which maps drifted concepts to the space of known concepts.

Without loss of generality, we assume the number of images in each data chunk to be the same. Let $X^{(t)} \in R^{D\times N}$ be the data chunk arrived at time t where D and N denote the dimensionality of the image descriptor and the number of images in a chunk, respectively. In this work, the image database is firstly normalized to center at the origin of the input space. At time $t=0$, the first data chunk provides the initial image set of each concept including images and corresponding labels. Therefore, the task of CPH at $t=0$ is to learn a set of hash functions to distinguish those given concepts. Hashing methods aim to learn a set of hash functions such that images of the same concept will have similar hash codes while images of different concepts have dissimilar hash codes. In this way, the similarity information of the images in this data chunk is preserved. In addition, each hash function is also expected to divide images into different binary hash values evenly to maximize the entropy of hash bit [5]. Therefore, at $t=0$, CPH generates a set of B hash functions, $H^{(0)}=\{h_1, h_2, ..., h_B\}$, which maximizes both the similarity preservation and the variance of hash codes. This target is exactly same with the idea of the existing BSPLH. To learn the hash code well, we employ BSPLH in a supervised manner. After the training of hash functions, the hash code value of the b^{th} hash function (h_b) for an image x is computed as follows:

$$h_b(x) = sgn(w_b^T x) \tag{1}$$

where $sgn(\bullet)$ and w_b denote the sign function and the hash projection vector, respectively.

The key component of CPH is the concept preservation training over time after time $t = 0$. This is achieved by minimizing the hash code differences between images of the same concept at $t = 0$ and at the current time. After $H^{(0)}$ is learned, a set of n images from each semantic class is randomly drawn from the $X^{(0)}$ as concept reference images. Projected hash vectors of concept reference images are used as the representation of each existing concept. To provide a projection reference for the following hashing projections after $t = 0$, we record the position of these images after the original hash projections. To improve the accuracy, the real-valued hash function outputs (by Equation 1 without the use of sign function) of concept reference images are recorded as the Concept Reference Matrix $\Psi \in R^{nC\times B}$, where C denotes the number of concepts in the data chunk. This matrix records the distribution of images in each concept after hashing projection. Thus, in the following training process of CPH, the newly generated hash codes are expected to map to the corresponding area of the same concept.

After the set of hash functions is trained for the data chunk at time t, hash codes for newly arrived images are computed by $H^{(t)}$ and then added to the database storing hash codes of images arrived since $t = 0$. Hash codes of the query image arrived at time t are computed using the latest K sets of hash functions because query images are drawn from the current data environment. Then accumulated Hamming distance between K hash codes of the query and the hash codes of all images in the database are computed. Images in the database with hash codes yielding minimum accumulated Hamming distances from the hash code of the query are returned as the image retrieval results. CPH is illustrated in a schematic diagram as shown in Figure 1, where two consecutive time steps are considered. At time $t = T$, data distributions of existing concepts A and B change and a new concept C emerges. By CPH, new images of existing concepts A and B are expected to be projected to the same areas in the hash projected space for old images belonging to the same concepts, A or B. The Concept Reference Matrix consists of projection references for concepts A and B, but has no information for the new concept C when it emerges at $t = T$. Thus, new images of new concept C are projected to an area far from the areas for other existing concepts in the hash projected space. After the training of new hash functions, projection references of the new concept C will be added to Ψ which makes C to become an existing concept in $t > T$. In this way, the Concept Reference Matrix Ψ records the projection references for all concpets. Newly appearing images are projected to be close to projection references of its concpet without updating existing hash codes of old images. CPH generates hash codes for newly appearng images only and avoids frequent access to old images which is very practical for real world applications. To our best knowledge, CPH is the first hashing method for non-stationary data environment which learns new hash function without updating hash codes of old images.

The details of the Concept Reference Matrix and the

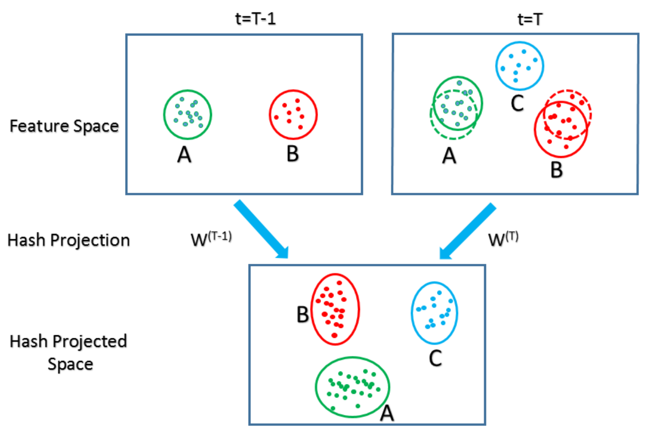


Fig. 1: The schematic diagram of CPH.

objective function of CPH to learn new hash functions are introduced in Sections III-A and III-B, respectively. Section III-C will derive the optimization method of CPH for hash codes generation. Section III-D introduces the retrieval method when the query image appears.

A. *Concept Reference Matrix*

As the major component of CPH, knowledge representation of the original concepts plays an important role. Similar to human mind, some examples of a concept in the hash projected space will be remembered to serve as the reference of this concept for the learning of following hash projections. New images with hash codes similar to the reference of one concept will be treated as the same concept. Borrowing this idea, CPH uses a Concept Reference Matrix to store examples of hash function values of images in different concepts.

For each class in the first data chunk at time $t = 0$, n images are randomly selected to form a set of nC reference images (i.e. $X^* \in R^{D\times nC}$) for a C-class problem. To provide more accurate information for concept preservation, real-valued hash function values are stored instead of the binary hash codes to construct the Concept Reference Matrix. Then, the Concept Reference Matrix $\Psi \in R^{nC\times B}$ is computed as follows:

$$\Psi = (X^*)^T W^{(0)} \tag{2}$$

In the case that the number of images in a concept class is less than n, then all images of this concept class are used as reference images. At the next time step, extra reference examples of this concept class will be randomly selected until there are n example images of this concept class added to X^*. Then, their hash function values computed using the corresponding new hash functions will be added to Ψ. In the case that new concept appearing at time $t > 0$, n images of this concept will also be added into X^*.

B. *Objective Function of CPH*

The objective of CPH is to preserve concept by projecting new images of the same concept to the same set of hash codes while maintaining a high accuracy in distinguishing images of different concepts. To train a new set of hash functions when a new data chunk appears, three aspects are taken into consideration: isomorphic similarity, hash codes partition balancing, and heterogeneous similarity fitness. Therefore, CPH is set to maximize the following objective function:

$$\Phi(W) = J(W) + \alpha J^*(W) + \beta tr\{W^T X X^T W\} \tag{3}$$

where α and β are non-negative parameters of CPH; X and W denote the set of training images in the newest data chunk and the current hash projection matrix to be learned, respectively. CPH maximizes the objective function in Equation 3 to find the optimal hash functions (i.e. W).

In Sections III-B and III-C, the focus is on works done within each time step therefore the superscript t for time step is dropped for simplicity. The first component $J(W)$ preserves the isomorphic similarity of images in the current data chunk while $J^*(W)$ preserves the similarity between the images of original concepts and the concepts present in the current data chunk, i.e. heterogeneous similarity fitness. The last component enhances the variance of the hash codes being learned to yield higher entropy. Instead of directly maximizing the differences among hash codes of images, it is relaxed to maximize the variance of hash values i.e. $W^T X X^T W$ as in [5]. Components $J(W)$ and $J^*(W)$ are derived in the following paragraphs.

Two images are regards as similar when they share the same label. Using the concept class labels of images in the current data chunk, the element in pairwise similarity matrix S is computed as follows:

$$S_{ij} = \begin{cases} +1, & x_i \text{ and } x_j \text{ are similar} \\ -1, & x_i \text{ and } x_j \text{ are dissimilar} \end{cases} \tag{4}$$

where x_i and x_j are two images in the current data chunk. For the case that each image having multiple labels, the number of common labels can be used as the pairwise similarity between two images. To remove the need of computing similarity for the same image, we set diagonal elements in S to zero. The similarity of concepts of the current data chunk, i.e. isomorphic similarity, can be preserved by finding hash codes maximizing the quantity in Equation 5 as follows:

$$\begin{aligned} J(H) &= \sum_{b=1}^{B} \{ \sum_{\forall S_{ij}=1} h_b(x_i)h_b(x_j) - \sum_{\forall S_{ij}=-1} h_b(x_i)h_b(x_j) \} \\ &= tr\{H(X)SH(X)^T\} \\ &= tr\{sgn(W^T X)Ssgn(W^T X)^T\} \end{aligned} \tag{5}$$

Then, Equation 5 can be relaxed by removing the $sgn(\bullet)$ to obtain the following:

$$J(W) = tr\{W^T X S X^T W\} \tag{6}$$

The major challenge of dealing with image retrieval problems with concept drift is that images of the same concept may locate in different regions of the input feature space at different time t (i.e. concept drift). Stationary hashing methods fail to deal with concept drift by hashing images of the same concept to different hash codes at different t. Therefore, CPH adds a new component (i.e. heterogeneous similarity fitness) to the objective function which maximizes the fitness of hash values of new data with the projection references of the same concept in Ψ, as follows:

$$\max J^*(W) = tr\{W^T X S^* \Psi\} \tag{7}$$

where S^* is the pairwise similarity matrix between X and X^* and is not symmetric. So, the objective function of CPH can be written as:

$$\begin{aligned} \Phi(W) = {} & tr\{W^T X S X^T W\} + \alpha tr\{W^T X S^* \Psi\} + \\ & \beta tr\{W^T X X^T W\} \end{aligned} \tag{8}$$

In the case of new concept appearing, since all projection references in Ψ belong to different concepts from this new concept, the heterogeneous similarity fitness, i.e. Equation 7, will provide dissimilarity information between new and old concepts. In this way, images of this new concept will be projected to an area in the hash projected space which is different from those of other concepts. Then, after the training of new hash functions W, hash projected values of reference images of this new concept will be added into Ψ. In later time steps, this concept will be preserved based on Ψ.

C. Hash Codes Generation by CPH

At time $t = 0$, Ψ does not exist. Therefore, we set $\alpha = 0$ and the objective function of CPH becomes:

$$\Phi(W) = tr\{W^T X S X^T W\} + \beta tr\{W^T X X^T W\} \tag{9}$$

This objective function is similar to the one in BSPLH [8] and thus the sequential learning procedure of BSPLH is used to learn the B-bit hash functions. At $t > 0$, Ψ is built and the full objective function of the CPH is maximized to find hash functions for the new data chunk. Based on the newest data chunk and Ψ, the optimal W is computed by setting the partial derivative of $\Phi(W)$ respect to W to be zero, as follows:

$$\begin{aligned} & \frac{\partial \Phi(W)}{\partial W} = 0 \\ & \Rightarrow (XSX^T + \beta X X^T)W + \alpha X S^* \Psi = 0 \end{aligned} \tag{10}$$

which is a simple linear regression problem, and W is directly computed as follows:

$$W = -\alpha(XSX^T + \beta XX^T)^{-1} X S^* \Psi \tag{11}$$

Based on Equation 11, new hash functions in CPH can be computed directly based on the concept reference matrix and newly appearing images which is very efficient, especially new hash functions are trained at each time step. Equation 11 requires $(XSX^T + \beta XX^T)$ to be positive definite for its inverse to exist, so we give the sufficiency conditions for it in the following.

Lemma 1: For a function $f(x) = y(x) + g(x)$, we have $\min\{f(x)\} \geq \min\{y(x)\} + \min\{g(x)\}$.

Lemma 2: $\min(x^T Ax) = \lambda^A_{\min}$ if $A \in R^{m\times m}$, $x \in R^{m\times 1}$, and $x^T x = 1$ where m and $\lambda^A_{\min}$ denote the dimension of the variables x and the smallest eigenvalue of A, respectively.

Based on these two lemmas, we have:

Proposition 1: The matrix $U = Y + \beta Z$ is positive definite if $\lambda^Y_{\min} + \beta\lambda^Z_{\min} > 0$ and $\beta > 0$, where $Y = XSX^T$, $Z = XX^T$.

$\lambda^Y_{\min}$ and $\lambda^Z_{\min}$ denote the smallest eigenvalue of the matrix Y and Z, respectively.

Proof 1: If U is positive definite, then U must satisfy

$$\begin{aligned} &\forall x \neq 0, x^T U x > 0 \\ \Leftrightarrow\ &\forall x \neq 0, x^T (Y + \beta Z) x > 0 \\ \Leftrightarrow\ &\forall x^T x = 1, \min\{x^T (Y + \beta Z) x\} > 0 \end{aligned} \tag{12}$$

Given that $x^T (Y + \beta Z) x = x^T Y x + \beta x^T Z x$ and based on Lemma 1, we have:

$$\min\{x^T (Y + \beta Z) x\} \geq \min\{x^T Y x\} + \min\{\beta x^T Z x\} \tag{13}$$

By combining Lemma 2, 12, and 13, we have:

$$\min\{x^T Y x\} + \beta \min\{x^T Z x\} = \lambda^Y_{\min} + \beta \lambda^Z_{\min} > 0 \tag{14}$$

Therefore, we can conclude that $\lambda^Y_{\min} + \beta \lambda^Z_{\min} > 0$ and $\beta > 0$ are sufficiency conditions for $(XSX^T + \beta XX^T)$ to be positive definite. Q.E.D.

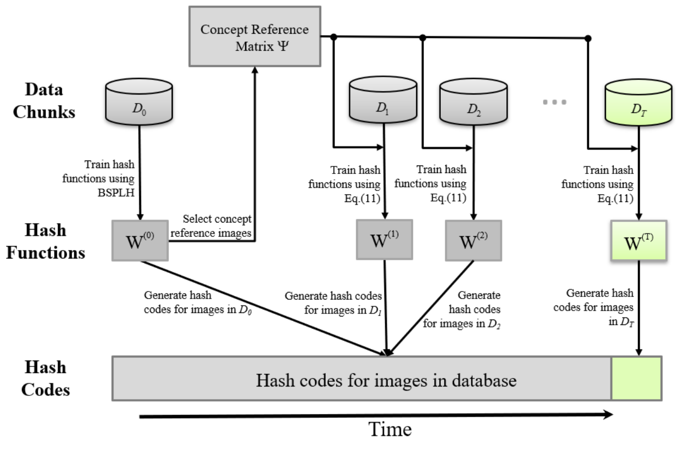
Proposition 1 shows the sufficiency condition for the term $(XSX^T + \beta XX^T)$ to be positive definite. However, the data set X and pairwise similarity matrix S are different at different time steps. The eigenvalues of the matrix XSX^T and XX^T are required to calculate the sufficiency condition. It is inefficient to check the sufficiency condition at each time step, thus in our experiments, a fixed value is selected for parameter β. In cases where the inverse of $(XSX^T + \beta XX^T)$ does not exist, the pseudo-inverse of the matrix will be used to compute Equation 11. Moreover, influences of different parameter values to the final retrieval performance are analyzed in Section IV-F. Given the fact that the precision of the CPH relies on the precision of concept learning in $t = 0$, a more time consuming but accurate sequential learning i.e. BSPLH is applied at $t = 0$. Afterward, a very fast direct computation of weight is applied for $t > 0$ to provide fast concept learning. Therefore, the overall training time of CPH is very fast, especially for large t in comparison to other online hashing methods.

D. Retrieval When Query Appears

In CPH, each image in the database has a unique hash code generated by its corresponding set of hash functions. In other words, the hash codes in the hash tables are generated by different sets of hash functions. Each set of hash functions corresponds to one data distribution at a time step. For a query image at time t, we need to generate the hash code for this query. The newest K sets of hash functions are used to generate K hash codes for this query because the query is drawn from the recent data environment. Then the summation of the Hamming distances between the K hash codes of the query and the unique hash code of a given image in the database is calculated as the final accumulated Hamming distance between the query and the given image. The images in the database with smallest final accumulated Hamming distance are returned as the retrieval results. When a query image x_q appears at time $t = T$, the final accumulated Hamming distance between x_q and image x_i is computed as follows:

$$D_{ham}(x_q, x_i) = \sum_{t=T-K+1}^{T} H^{(t)}(x_q) \oplus H^{(t)}(x_i) \tag{15}$$

where $D_{ham}(x_q, x_i)$, $\oplus$, and $H^{(t)}(x_i)$ denote the final accumulated Hamming distance function between two images, the XOR operation between hash codes, and the hash code of the image x_i, respectively. When there are not enough sets of hash functions in storage, i.e. $T < K$, all existing sets of hash functions are used to generate the hash codes of the query. Figure 2 shows an overview of CPH. A new data chunk appears at each time step. At time $t = 0$, hash functions $W^{(0)}$ are learned using BSPLH firstly. Concept reference images are then selected to generate the Concept Reference Matrix Ψ for the following training of hash functions. Hash codes for existing images are also generated and stored. In the following time steps, based on the Concept Reference Matrix and newly appearing images, new hash functions are learned at each time step using Equation 11. Meanwhile, hash codes of newly appearing data are also generated and stored. In Figure 2, the newest data chunk, learned hash functions and generated hash codes are marked with green for readability.



Concept Reference Matrix Ψ
Data Chunks
D_0
D_1
D_2
D_T
Train hash functions using BSPLH
Select concept reference images
Train hash functions using Eq.(11)
Train hash functions using Eq.(11)
Train hash functions using Eq.(11)
Hash Functions
$W^{(0)}$
$W^{(1)}$
$W^{(2)}$
$W^{(T)}$
Generate hash codes for images in D_0
Generate hash codes for images in D_1
Generate hash codes for images in D_2
Generate hash codes for images in D_T
Hash Codes
Hash codes for images in database
Time

Fig. 2: An Overview of CPH.

IV. Experimental Studies

Image retrieval in non-stationary environment is still a new research topic. As far as we know, there is no public non-stationary image database with concept drift problem available yet. In our experiments, two public real world image databases, CIFAR10 and CIFAR100, are utilized to simulate the non-stationary data environments. There are totally 11 non-stationary data scenarios being simulated to evaluate the retrieval performance of CPH and the comparative methods. In our experiments, CPH is compared with LSH [6], SH [21], BSPLH [8], OKH [9], OSH [10], OH [11] and ICH [12]. LSH and SH are representative unsupervised hashing methods and are used as baseline comparative methods. BSPLH is a representative semi-supervised hashing method which is also applied in the primary training procedure of CPH. The supervised OKH, OH and the unsupervised OSH both handle the image retrieval problem in non-stationary data environment, which are the main comparative methods in the study of CPH. ICH exactly handles the image retrieval problem in non-stationary environment with the concept drift problem. Thus, ICH is also a main comparative method used in this paper.

In our experiments, the parameters of each comparative method are selected as used in the original papers or official release of MATLAB libraries. For the proposed method CPH, two parameters involved in the objective functions are set as $\alpha = 1$ and $\beta = 0.05$ in our experiments. Moreover, the number of concept reference images selected in each concept and the number of hash codes generated generated for the query are set as 100 and 10, respectively. The value of thse parameters are anaylzed and selected in Section IV-F in detail. For the case that there are not enough images for the selection of concept reference images at time step $t = 0$, all existing images of this concept will be selected, and extra images in the following time steps will be selected until 100 concept reference images in total of this concept are found. For all employed hashing methods, the length of hash code is set to 64. For each experiment, 10 independent runs are performed and their average results are presented.

Mean Average Precision (MAP) and Top 1% Precision are used in our experiments as metrics to evaluate different hashing methods based on the semantic ground truth. MAP is a widely used evaluation metric which considers both the precision and ranks of the true positive returned samples. The Top 1% precision concept is firstly proposed in [12], which is based on the idea of Top N precision. Evaluating precision in an adaptive range, not a certain number N, is appropriate for evaluation in non-stationary environment.

11 simulated data scenarios of non-stationary environments with concept drift used in the experiments are introduced in Section IV-A. Experimental results of different hashing methods are presented and discussed in Section IV-B. In Section IV-C, CPH is compared with several modified stationary hashing methods to show its outstanding retrieval performance. Section IV-D compares the training times of different hashing methods. In Section IV-E, CPH is validated on multi-labeled images retrieval task. Parameters influencing the performance of CPH are investigated in Section IV-F.

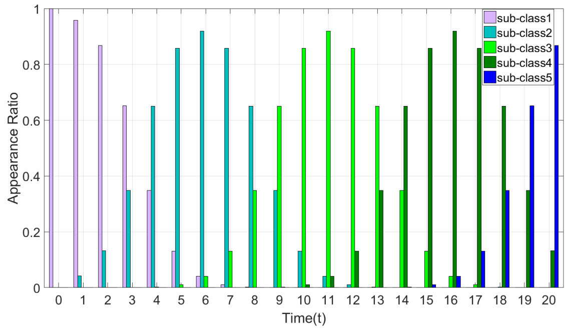
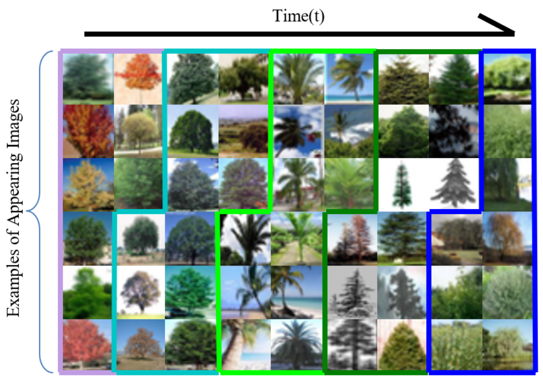


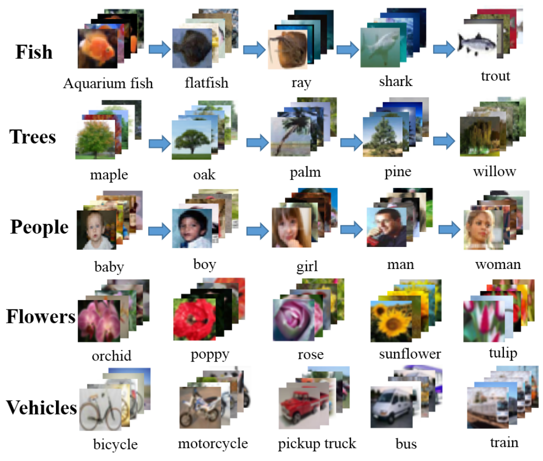
Fig. 3: Appearance ratio of five sub-classes in each super class.

A. Databases of Non-stationary Scenarios with Concept Drift

Two databases, CIFAR10 and CIFAR100, are used in our experiment to simulate 11 different non-stationary data scenarios. The CIFAR10 database consists of $60,000$ real world color images belonging to 10 semantic classes, such as airplane, cat, dog, and so on. Each class has $6,000$ images. Each image has 32×32 pixels and is described by a 512-dimensinal GIST descriptor. The CIFAR100 database consists of $60,000$ 32×32 color images belonging to 100 classes. Each class has 600 images, and each image is described by a 512-dimensional GIST feature vector. Moreover, 100 classes can be further grouped into 20 super classes. In other words, five similar but different smaller classes can be grouped into a larger common class. For example, the super class Trees consists of five smaller classes, i.e. *maple*, *oak*, *palm*, *pine*, and *willow*.



(a) Examples of appearing images for super class *Trees* over time.



(b) Examples of distribution drifting for five super classes.

Fig. 4: Examples of distribution drifting by controlling the appearance of sub-classes.

In this paper, two concept drift problems are focused: the distribution drifting and new class appearing, following the idea in [12]. 11 data scenarios are simulated in our experiments which consists of 5 distribution drifting data scenarios, 3 new class appearing scenarios, and 3 more complicated combined scenarios. The detailed settings of these 11 data scenarios are listed in tables I, II, and III. The CIFAR10 database is used to simulate the new class appearing data scenario, as shown in table I. Images randomly selected from 5 randomly selected classes are used as the original set from time 0 5. Then when $t >= 6$, images from the original classes and a certain number of new classes (i.e. 1, 3, and 5) are selected to form the following data chunks. Table II shows the settings of distribution drifting scenarios. The CIFAR100 database, which contains 20 super classes, is used. Each super class consists

TABLE I: Settings of simulation for new class appearing scenarios

Concept drift problem	$0 \leq t \leq 5$	$t >= 6$	Scenario Name
New class appearing	Images randomly selected from 5 classes	Images randomly selected from the original classes and 1 new class	CIFAR10_1
	Same as above	Images randomly selected from the original classes and 3 new classes	CIFAR10_3
	Same as above	Images randomly selected from the original classes and 5 new classes	CIFAR10_5

TABLE II: Settings of simulation for distribution drifting scenarios

Concept drift problem	$t = 0$	$t > 0$	Scenario Name
Distribution drifting	Images randomly selected from the first sub-class in 20 super classes	Subclasses of all existing super classes change according to Figure 3	CIFAR100_20
	Images randomly selected from the first sub-class in 15 super classes	Same as above	CIFAR100_15
	Images randomly selected from the first sub-class in 10 super classes	Same as above	CIFAR100_10
	Same as above	Subclasses of 6 existing super classes change according to Figure 3	CIFAR100_10_6
	Same as above	Subclasses of 3 existing super classes change according to Figure 3	CIFAR100_10_3

TABLE III: Settings of simulation for combined scenarios

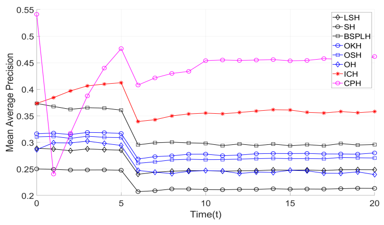
Concept drift problem	$0 \leq t \leq 20$	$21 \leq t \leq 40$	Scenario Name
Combined (new class appearing and distribution drifting happen simultaneously)	• 5 stationary super-classes • 5 super classes drift	10 new super classes appear	CIFAR100_DN
	• 15 super-classes without drift • 5 super classes appear as new classes since $t = 6$	• 10 out of 15 super classes drift • 5 super classes appear continuously without drift	CIFAR100_ND
	• 5 super classes without drift • 5 super-classes drift • 10 new super-classes appear at $t = 6$	Experiments end at $t = 20$	CIFAR100_D&N

of 5 smaller classes (also called sub class for simplicity). The distribution drifting phenomenon is simulated by controlling the appearance ratio of each sub class in the corresponding data chunk, as shown in Figure 3, in which the ratio values are generated by Gaussian functions with different mean values. In this way, the distribution of the super classes changes over time. Figure 4 shows examples of distribution drifting by controlling the appearance ratio of sub-classes. Figure 4(a) shows examples of appearing images for super class Trees over time, which consists of five sub classes with appearance ratio changes according to Figure 3. Each column represents example images appearing in a time step. Only 9 time steps are shown in this figure for simplicity. According to Figure 4(a), images belonging to different sub classes are visibly similar with little difference. Moreover, the appearance ratio of each sub class changes over time. In this way, the distribution of the super class *Trees* drifting over time is simulated. Figure 4(b) shows examples of distribution drifting happening in five super classes, i.e. *Fish*, *Trees*, *People*, *Flowers*, and *Vehicles*. Moreover, we further evaluate the proposed CPH on three more complicated data scenarios, namely CIFAR100_DN, CIFAR100_ND, CIFAR100_D&N, as shown in Table III. In CIFAR100_DN and CIFAR100_ND, the new class appearing scenario and distribution drifting scenario happens in different order. In CIFAR100_D&N, two concept drift scenarios happen simultaneously.

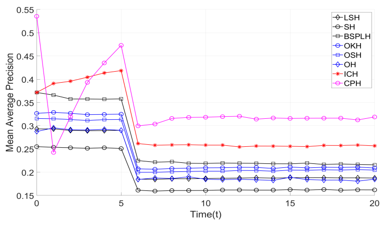
There are 21 data chunks being generated from $t = 0$ to 20 for all data scenarios, except data scenarios CIFAR100_DN and CIFAR100_ND in which two concept drift scenarios happen sequentially. CIFAR100_DN and CIFAR100_ND both have 41 data chunks from $t = 0$ to 40. At each time step t, both the training set and the testing set are generated by randomly selecting 1000 images from these classes according to the above appearance ratios. Moreover, 100 images in training set at each time step are randomly selected as the labeled images while all other images are regarded as unlabeled images for semi-supervised methods, i.e. BSPLH and ICH. For supervised hashing methods, OKH, OH and CPH, all images in the training set are used as labeled images. For OKH and OH, which learns image pairs in an online manner, the 1000 images in the training set are partitioned into 500 pairs with similarity information for training. For CPH, we set $n = 100$, $K = 10$, $\alpha = 1$, and $\beta = 0.05$. For the unsupervised OSH, all label information is ignored.

B. Experimental Results

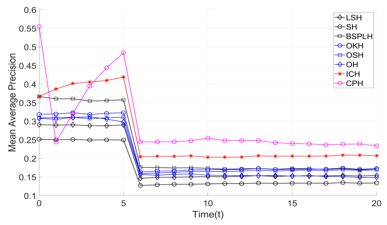
CPH is compared with representative stationary hashing methods such as LSH, SH, BSPLH and existing non-stationary hashing methods such as OKH, OSH, OH, and ICH. Comparisons are performed on 11 non-stationary data scenarios.



(a) CIFAR10_1

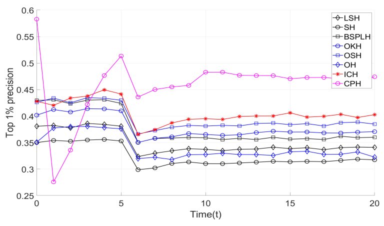


(b) CIFAR10_3

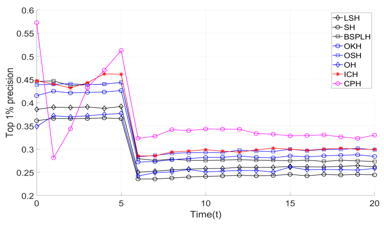


(c) CIFAR10_5

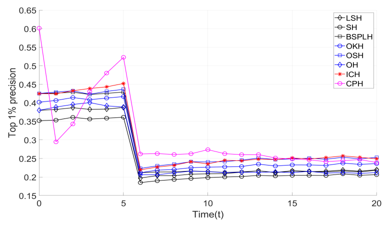
Fig. 5: MAP of CPH and comparative hashing methods on 3 new class appearing data scenarios.



(a) CIFAR10_1

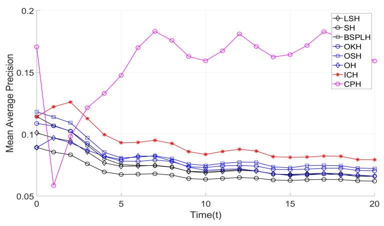


(b) CIFAR10_3

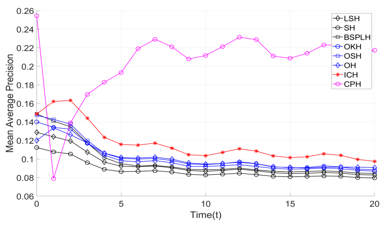


(c) CIFAR10_5

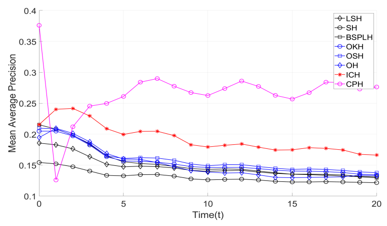
Fig. 6: Top 1% precision of CPH and comparative hashing methods on 3 new class appearing data scenarios.



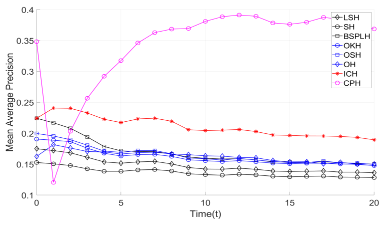
(a) CIFAR100_20



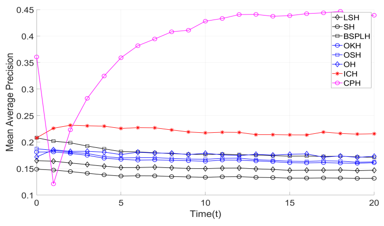
(b) CIFAR100_15



(c) CIFAR100_10



(d) CIFAR100_10_6



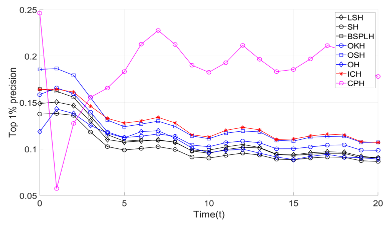
(e) CIFAR100_10_3

Fig. 7: MAP of the CPH and comparative hashing methods on 5 distribution drifting data scenarios.

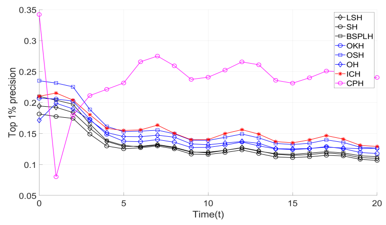
Section IV-B1 introduces the experimental results on the new class appearing data scenarios. Comparisons on the distribution drifting data scenarios are shown in Section IV-B2. Section IV-B3 shows the experimental results on three more complicated data scenarios with settings listed in Table III.

1) Comparisons on New Class Appearing Data Scenarios: The experimental results in terms of MAP and Top 1% precision for all comparative methods and CPH are shown in Figures 5 and 6, respectively.

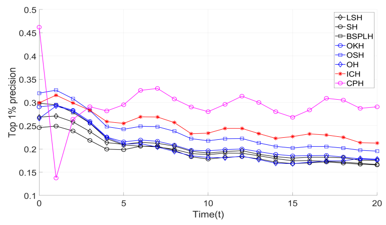
In both Figure 5 and Figure 6, the unsupervised LSH and SH obtain the lowest retrieval accuracy, since no semantic information is used for training. Moreover, hash functions of these two methods are trained at the beginning and never update. OSH and OKH achieve lower MAP but higher top 1% precision than BSPLH. This suggests that there are more correct images being returned in the top 1% image set with least Hamming distance for OKH and OSH than BSPLH. However, the rank of these correctly returned images is relatively higher than BSPLH. In these new class appearing data scenarios, the unsupervised OSH updates hash functions adaptively without considering the semantic similarity information. OKH and OH updates hash functions with pairwise similarities of two labels, while BSPLH utilizes the similarities between all pairs of data in the data chunk. For a data chunk containing n labeled images, only $n/2$ pairwise similarities will be used for training in OKH and OH, while C_n^2 pairwise similarities could be used for training in BSPLH. This is a big drawback of OKH and OH which is also the reason that in some cases (e.g. Figure



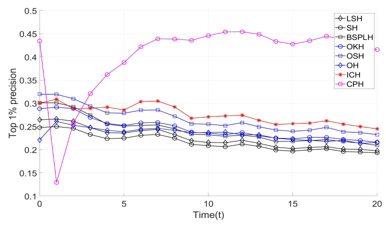
(a) CIFAR100_20



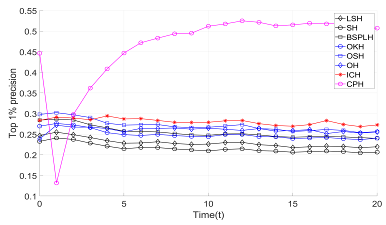
(b) CIFAR100_15



(c) CIFAR100_10

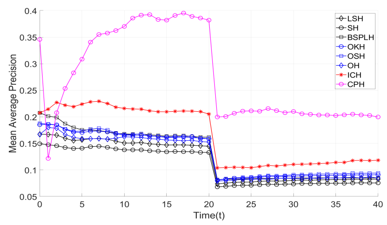


(d) CIFAR100_10_6

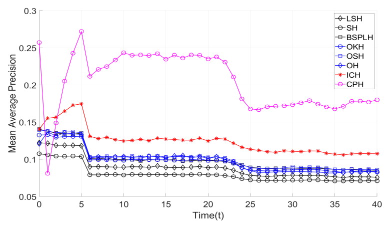


(e) CIFAR100_10_3

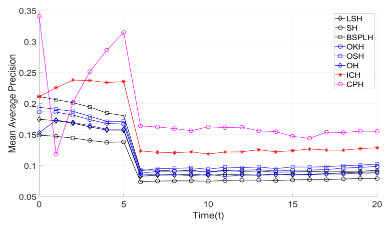
Fig. 8: Top 1% precision of CPH and comparative hashing methods on 5 distribution drifting data scenarios.



(a) CIFAR100_DN

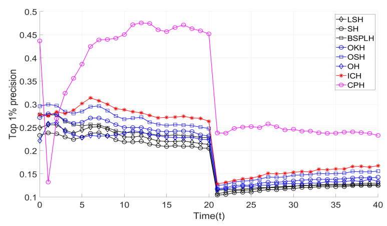


(b) CIFAR100_ND

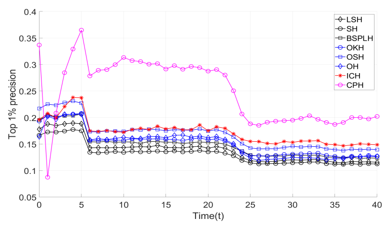


(c) CIFAR100_D&N

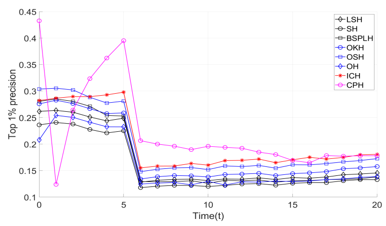
Fig. 9: MAP of CPH and comparative hashing methods on 3 combined data scenarios.



(a) CIFAR100_DN



(b) CIFAR100_ND



(c) CIFAR100_D&N

Fig. 10: Top 1% precision of CPH and comparative hashing methods on 3 combined data scenarios.

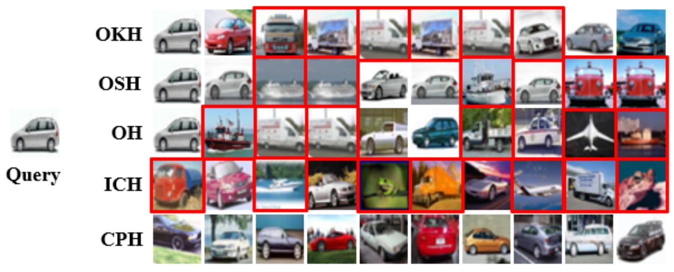
5), the supervised OKH and OH cannot achieve better retrieval performance than the stationary BSPLH. CPH achieves highest MAP and Top 1% precision among all hashing methods in comparison for 3 new class appearing scenarios. The big drop of retrieval performance for CPH at time $t = 1$ is caused by the change from using sequential learning to direct computation of W for hash function generation after $t > 0$. Even with this drop, the performance of CPH recovers quickly and keeps the highest performance level comparing with other hashing methods.

2) Comparisons on Distribution Drifting Data Scenarios: Figure 7 and Figure 8 show MAP and Top 1% precision of CPH and comparative methods on 5 distribution drifting data scenarios, respectively. In the first three data scenarios, different number (i.e. 20, 15, 10) of super classes are utilized to simulate the distribution drifting situation. Afterwards, 10 super classes are used for simulation in which 6 (and 3) super classes drifts and left classes keep unchanged.

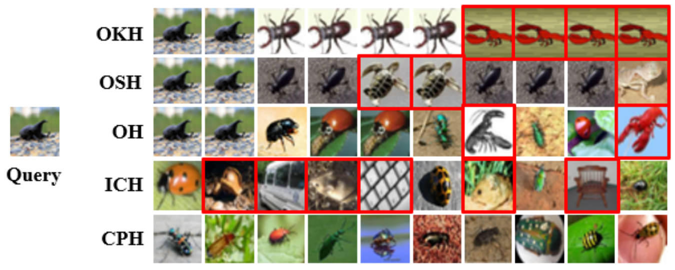
According to Figure 7 and Figure 8, CPH outperforms other hashing methods significantly. At the time step $t = 1$, there is significant performance drop for CPH, due to the change of hash functions training method. However, after time $t = 1$, the performance of CPH recovers quickly, just like the phenomenon shown in the new class appearing data scenarios. Moreover, the performance of CPH keeps stable finally at a high level with new data appearing over time. ICH employs multiple hash tables and adjusts the weights of hash tables adaptively, which achieves the second best retrieval performance. The representative online hashing methods OKH, OH and OSH achieve relatively lower MAP and Top 1% precision. But all three methods still perform better than stationary hashing methods, such as BSPLH, LSH, and SH.

3) Comparisons on Combined Data Scenarios: Figures 9 and 10 show the experimental results on 3 combined data scenarios. CPH outperforms all other comparative methods. In data scenario CIFAR100_DN, 5 super classes drift from time $t = 0$ 20, and 10 super classes appear as new concepts at time $t = 21$. Thus the performance of CPH has a significant drop at time $t = 21$. In data scenario CIFAR100_ND, 15 original super classes appear without drifting, and 5 super classes appear at time $t = 6$ as the new concepts. Then 10 of the 15 original super classes begin drifting. The performance of all hashing methods drops at time $t = 6$ due to the new class appearing. Moreover, after time $t = 21$, all hashing methods tend to drop for a period of time, since 10 super classes begin drift. After several time period, the retrieval performance of hashing methods become stable. In the CIFAR100_D&N data scenario, images from 5 stationary classes appear over time. 5 super classes drift during time $t = 0$ 20. Moreover, 10 new super classes appear at time $t = 6$ to simulate the new class appearing situation. In this data scenario, both concept drift problems happen simultaneously which makes this data scenario most complicated. In this data scenario, CPH also outperforms all other hashing methods.

Moreover, visual retrieval results of CPH and other four non-stationary hashing methods on two data scenarios, i.e. CIFAR10_1 and CIFAR100_10_3, at time $t = 20$ are shown in Figure 11. The query image is randomly selected from the query set at time $t = 20$. For the given query image, top 10 nearest images returned by each hash method are shown at the corresponding row. Wrongly returned images which are dissimilar to the query image are marked with red rectangles. Some returned images are the same. That is because images in data chunks at different time steps are selected randomly and independently from the original image set. Therefore, image may be selected more than once. According to Figure 11, the proposed CPH method achieves significantly outstanding retrieval performance comparing to other methods.



(a) Visual retrieval results on CIFAR10_1 data scenario.



(b) Visual retrieval results on CIFAR100_10_3 data scenario.

Fig. 11: Visual retrieval results of non-stationary hashing methods on two data scenarios.

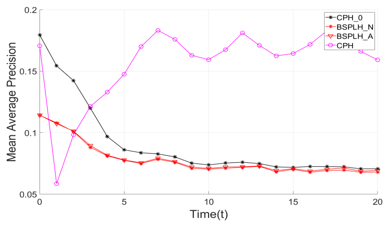
C. Comparison with Modified Existing Hashing Methods

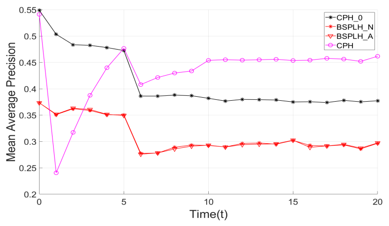
CPH achieves a good retrieval performance at the beginning in all data scenarios. To further evaluate CPH with updating, it is compared with a static version of CPH which trains hash functions at the beginning by CPH but does not update hash functions afterwards. This version of CPH is named as CPH_0. Moreover, in this section, we also compare CPH with two simply modified versions of the existing stationary hashing methods, to show the performance improvement of CPH. Since BSPLH is utilized to train the first set of hash functions in CPH with all data labeled, both two comparative methods, i.e. BSPLH_N and BSPLH_A, are based on BSPLH. BSPLH_N re-trains hash functions by BSPLH at each time step utilizing the newest data chunk. BSPLH_A re-trains hash functions at each time step by using all existing labeled and unlabeled data. Figure 12 shows the MAP results of CPH, CPH_0 and two modified methods on three kinds of non-stationary data scenarios: distribution drifting scenario CIFAR100_20, new class appearing scenario CIFAR10_1, and combined scenario CIFAR100_D&N.

BSPLH_N re-trains hash functions only using the newest data chunk and ignores the previous data information. BSPLH_A utilizes all existing data for training directly, while the newest data distribution may be different with the previous ones. According to Figure 12, BSPLH_N and BSPLH_A achieve nearly the same MAP performance. Moreover, their MAP performance is the lowest, comparing to CPH and CPH_0. Therefore, simply updating hash functions by the newest chunk of data or directly combining all existing data for training is not able to generate suitable hash functions for image retrieval task in non-stationary data environment. CPH_0 achieves outstanding retrieval performance at the beginning. Since it does not update hash functions afterwards, CPH_0 cannot adapt to the new data environment when concept drift happens. As a result, the retrieval performance of CPH_0 degrades significantly later. CPH achieves the best performance in all three data scenarios which generate new hash functions to preserve the similarity between new and old images in different times.

D. Comparison in Training Time

In this section, we compare the training times of different hashing methods for the experiment using the CIFAR100_20 scenario. Figures 13 and 14 show the individual training time at each time step and the accumulated training time over time of each hashing method, respectively. CPH yields the largest individual training time at $t = 0$, due to the fully supervised training process of BSPLH. However, the individual training time for training new hash functions is very little. Thus,


(a) CIFAR100_20


(b) CIFAR10_1

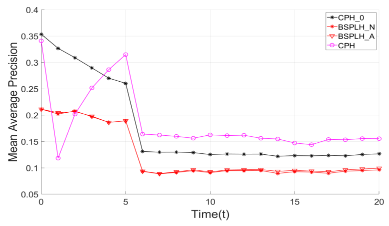

(c) CIFAR100_D&N

Fig. 12: MAP of CPH, CPH_0 and two modified versions of BSPLH.

CPH is efficient to train new hash functions with new data appearing over time. Moreover, its accumulated training time is dominated by the training time at $t = 0$. It is obvious that in a long run, small individual training time at $t > 0$ is a very important for large scale image retrieval problems.

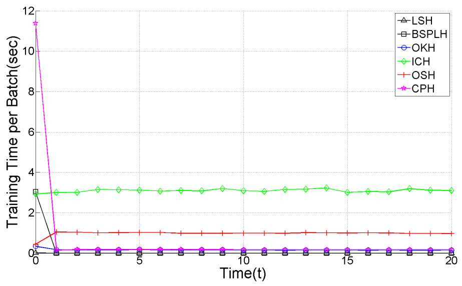


Fig. 13: Individual training time of different hashing methods at each time step.

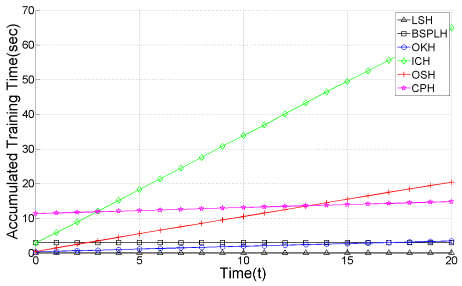


Fig. 14: Accumulated training time of different hashing methods at each time step.

ICH is a semi-supervised incremental hashing method. It takes the second longest individual training time at time $t = 0$ and it is almost a constant in all time steps. This leads to the ICH yielding a very large accumulated training time over a long time period. OSH yields a relatively small individual training time, but the accumulated training time becomes very large after a period of time and increases steadily. The accumulated training time of OSH is larger than that of CPH when $t > 13$. LSH and BSPLH are stationary methods and never update after time $t = 0$. Thus their individual training time after time $t = 0$ is zero. The individual training time of OKH after time $t = 0$ is similar to the proposed CPH, which is also efficient. Given the significantly better retrieval precision and low individual training time at all $t > 0$, CPH is a very promising hashing method for non-stationary image retrieval problems with concept drift for a long run.

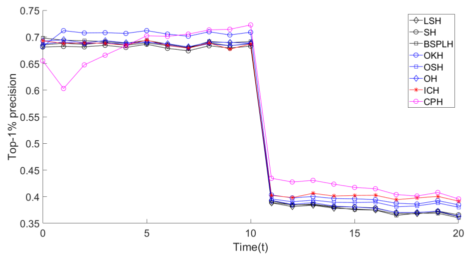
E. Image Retrieval for Multi-labeled Images

In the previous sections, CPH has shown its superior performance for single-labeled images. In this section, CPH is validated on a more challenging task, i.e. multi-labeled images retrieval. Real world image database NUSWIDE is employed which consists of $269,648$ images belonging to 81 categories. For multi-labeled images retrieval task, two images are regarded as similar when they share at least one same label. Each image is represented by a 500-dimension bag of words based on SIFT descriptors. This database is used to simulate the new class appearing scenario, in which 20 categories are randomly selected as the original classes. Images belonging to original classes are randomly selected to generate the data chunks from time $t = 0$ to 10. After time $t = 10$, images from all 81 categories are randomly selected to generate data chunks.

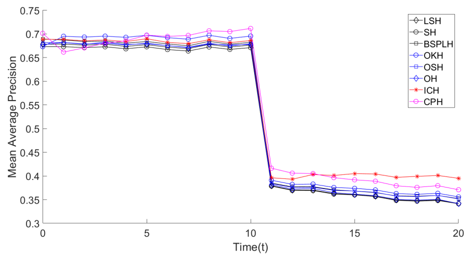
Top 1% precision and MAP of CPH and other comparative methods are shown in Figure 15. According to this figure, the stationary hashing methods LSH, SH, and BSPLH yield the worst retrieval performance after new class appearing since time $t = 11$. CPH achieves highest Top 1% precision and satisfying MAP performance just lower than ICH. Among ICH and CPH, ICH employs multiple hash tables with larger storage cost and updates hash codes of all images at each time step. CPH generates hash codes for new image only without updating hash codes of all images which is more efficient. Other non-stationary hashing methods OKH, OSH, and OH achieve better retrieval performance comparing to stationary hashing methods but lower than CPH's.

F. Parameter Selection

There are two sets of parameters need to be selected for CPH. The first set consists of two parameters: α and β for the objective function, while the second set consists of three parameters: the number of bits (B), the number of images in a concept class being selected for Ψ (n),and the number of hash codes generated for the query (K). Average Top 1% precisions over $t = 0$ to 20 for different parameter values for the CIFAR100_D&N scenario are used to evaluate the performance of parameter selection. Figure 16 shows the average precisions for different α and β values with $B = 64$,



(a) Top 1% precision



(b) MAP

Fig. 15: Top 1% precision and MAP of CPH and other comparative methods.

$n = 100$, and $K = 10$. According to this figure, the retrieval performance of the CPH method is insensitive to values of α and β. In experiments, $\alpha = 1$ and $\beta = 0.05$ are used because this set of parameters yields the best performance. With $\alpha = 1$ and $\beta = 0.05$, average Top 1% precisions over $t = 0$ to 20 for different K and n values for the CIFAR100_D&N scenario are shown in Figure 17. According to this figure, we choose $K = 10$ and $n = 100$ in all experiments. Figure 18 shows the Top 1% precision of CPH with different hash code length. When $B = 64$, CPH achieves almost the best performance, which is just lower than the case $B = 128$. Consider the training efficiency, we select $B = 64$ as the hash code length in all our experiments.

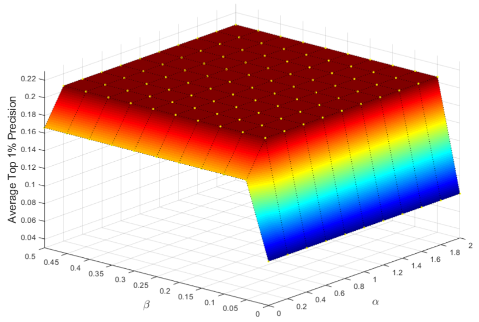


Fig. 16: Average top 1% precisions for different β and α values.

V. Conclusions

In this paper we propose a new method, i.e. *concept preserving hashing*, for image retrieval in non-stationary data environment with concept drift. With new data chunk appearing,

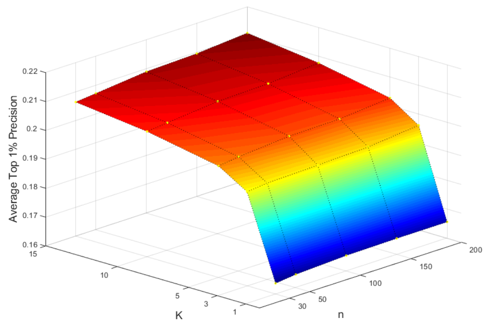


Fig. 17: Average top 1% precisions for different K and n values.

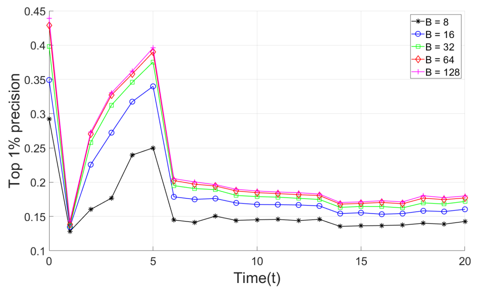


Fig. 18: Top 1% precisions of CPH with different hash code length B.

CPH generates hash codes for new images which are similar to the hash codes of previous images of the same concept. In other words, CPH projects images of the same concept from different times (i.e. different distributions) to a similar area in the hash projected space. In this way, there is no need to update the learned hash codes of previous images. To generate the objective function for learning new hash functions, three aspects are taken into consideration C isomorphic similarity, hash codes partition balancing, and heterogeneous similarity fitness. The new hash functions can be learned directly which is very efficient. As far as we know, this is the first work to handle image retrieval in non-stationary environment with concept drift without updating old hash codes. This attempt is very valuable and practical for large scale image retrieval in non-stationary environment.

There are also several drawbacks of CPH which need further improvement. One is that there is a significant performance drop at the beginning, which is caused by the different training procedure of hash functions. Thus, a further work is to strengthen the updating methods of hash functions to achieve more stable retrieval performance. Moreover, CPH trains new hash functions at each time step to adapt to new data environment like other non-stationary hashing methods, while concept drift may be slight or not happening in some time steps. Therefore, concept drift detection for deciding whether new hash functions need to be updated is another interesting topic. In recent years, transfer learning technique has been widely researched and achieved huge success, which utilizes knowledge of source domain to benefit the training on target domain. Applying transfer learning technique to preserve the concept when learning new hash functions could be a meaningful idea and worthy of further research. This is also one of our future works.

References

[1] B. Marr, *Big Data: Using SMART big data, analytics and metrics to make better decisions and improve performance*. Wiley, 2015.

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

[3] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*. IEEE, 2008, pp. 1–8.

[4] 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.

[5] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2393–2406, 2012.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*. ACM, 2004, pp. 253–262.

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

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

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

[10] C. Leng, J. Wu, J. Cheng, X. Bai, and H. Lu, "Online sketching hashing," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2503–2511.

[11] L.-K. Huang, Q. Yang, and W.-S. Zheng, "Online hashing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2309C–2322, 2018.

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

[13] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 769–790, 2018.

[14] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Advances in Neural Information Processing Systems*, 2009, pp. 1509–1517.

[15] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proceedings of the International Conference on Computer Vision*. IEEE, 2009, pp. 2130–2137.

[16] K. Jiang, Q. Que, and B. Kulis, "Revisiting kernelized locality-sensitive hashing for improved large-scale image retrieval," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4933–4941.

[17] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.

[18] X. Bai, H. Yang, J. Zhou, P. Ren, and J. Cheng, "Data-dependent hashing based on p-stable distribution," *IEEE Transactions on Image Processing*, vol. 23, no. 12, pp. 5033–5046, 2014.

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

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

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

[22] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 2957–2964.

[23] ——, "Spherical hashing: Binary code embedding with hyperspheres," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2304–2316, 2015.

[24] X. Liu, J. He, B. Lang, and S.-F. Chang, "Hash bit selection: A unified solution for selection problems in hashing," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*. IEEE, 2013, pp. 1570–1577.

[25] X. Liu, C. Deng, B. Lang, D. Tao, and X. Li, "Query-adaptive reciprocal hash tables for nearest neighbor search," *IEEE Transactions on Image Processing*, vol. 25, no. 2, pp. 907–919, 2016.

[26] F. Shen, C. Shen, Q. Shi, A. v. d. Hengel, Z. Tang, and H. t. Shen, "Hashing on nonlinear manifolds," *IEEE Transactions on Image Processing*, vol. 24, no. 6, pp. 1839C–1851, 2015.

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

[28] S. Wang, C. Li, and H.-L. Shen, "Distributed graph hashing," *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1896–1908, 2019.

[29] X. Bai, C. Yan, H. Yang, L. Bai, J. Zhou, and E. R. Hancock, "Adaptive hash retrieval with kernel based similarity," *Pattern Recognition*, vol. 75, pp. 136–148, 2018.

[30] X. Liu, Z. Li, C. Deng, and D. Tao, "Distributed adaptive binary quantization for fast nearest neighbor search," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5324–5336, 2017.

[31] H. Liu, R. Ji, J. Wang, and C. Shen, "Ordinal constraint binary coding for approximate nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 4, pp. 941–955, 2018.

[32] B. Kulis, P. Jain, and K. Grauman, "Fast similarity search for learned metrics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2143–2157, 2009.

[33] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua, "Ldahash: Improved matching with smaller descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 66–78, 2012.

[34] P. Li, M. Wang, J. Cheng, C. Xu, and H. Lu, "Spectral hashing with semantically consistent graph for image indexing," *IEEE Transactions on Multimedia*, vol. 15, no. 1, pp. 141–152, 2013.

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

[36] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, 2015, pp. 37–45.

[37] 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.

[38] C. Ma, I. W. Tsang, F. Shen, and C. Liu, "Error correcting input and output hashing," *IEEE Transactions on Cybernetics*, vol. 49, no. 3, pp. 781–791, 2019.

[39] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proceedings of the International Conference on Computer Vision and Pattern Recognition*. IEEE, 2016, pp. 2064–2072.

[40] H. Lai, P. Yan, X. Shu, Y. Wei, and S. Yan, "Instance-aware hashing for multi-label image retrieval," *IEEE Transactions on Image Processing*, vol. 25, no. 6, pp. 2469–2479, 2016.

[41] C. Deng, Z. Chen, X. Liu, X. Gao, and D. Tao, "Triplet-based deep hashing network for cross-modal retrieval," *IEEE Transactions on Image Processing*, vol. 27, no. 8, pp. 3893–3903, 2018.

[42] J. Cheng, C. Leng, P. Li, M. Wang, and H. Lu, "Semi-supervised multi-graph hashing for scalable similarity search," *Computer Vision and Image Understanding*, vol. 124, pp. 12–21, 2014.

[43] 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.

[44] 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.

[45] F. Cakir and S. Sclaroff, "Online supervised hashing," in *Proceedings of the International Conference on Image Processing*. IEEE, 2015, pp. 2606–2610.

[46] ——, "Adaptive hashing for fast similarity search," in *Proceedings of the International Conference on Computer Vision*, 2015, pp. 1044–1052.

[47] F. Cakir, K. He, S. A. Bargal, and S. Sclaroff, "Mihash: Online hashing with mutual information," in *Proceedings of the International Conference on Computer Vision*. IEEE, 2017, pp. 437–445.