Online Hashing

Long-Kai Huang, Qiang Yang, and Wei-Shi Zheng

***Abstract*—Although hash function learning algorithms have achieved great success in recent years, most existing hash models are off-line, which are not suitable for processing sequential or online data. To address this problem, this work proposes an online hash model to accommodate data coming in stream for online learning. Specifically, a new loss function is proposed to measure the similarity loss between a pair of data samples in hamming space. Then, a structured hash model is derived and optimized in a passive-aggressive way. Theoretical analysis on the upper bound of the cumulative loss for the proposed online hash model is provided. Furthermore, we extend our online hashing from a single-model to a multi-model online hashing that trains multiple models so as to retain diverse online hashing models in order to avoid biased update. The competitive efficiency and effectiveness of the proposed online hash models are verified through extensive experiments on several large-scale datasets as compared to related hashing methods.**

***Index Terms*—Hashing, online hashing**

I. Introduction

There have been great interests in representing data using compact binary codes in recent developments. Compact binary codes not only facilitate storage of large-scale data but also benefit fast similarity computation, so that they are applied to fast nearest neighbor search [1]–[4], as it only takes very short time (generally less than a second) to compare a query with millions of data points [5].

For learning compact binary codes, a number of hash function learning algorithms have been developed in the last five years. There are two types of hashing methods: the data independent ones and the data dependent ones. Typical data independent hash models include Locality Sensitive Hashing (LSH) [6] and its variants like ℓ_p-stable hashing [7], min-hash [8] and kernel LSH (KLSH) [9]. Since using information of data distribution or class labels would make significant improvement in fast search, more efforts are devoted to the data-dependent approach [10]–[16]. For the data dependent hashing methods, they are categorized into unsupervised-based [17]–[20], supervised-based [21]–[26], and semi-supervised-based [27]–[29] hash models. In addition to these works, multi-view hashing [30], [31], multi-modal hashing [32]–[36], and active hashing [37], [38] have also been developed.

In the development of hash models, a challenge remained unsolved is that most hash models are learned in an offline mode or batch mode, that is to assume all data are available in advance for learning the hash function. However, learning hash functions with such an assumption has the following critical limitations:

- First, they are hard to be trained on very large-scale training datasets, since they have to make all learning data kept in the memory, which is costly for processing. Even though the memory is enough, the training time of these methods on large-scale datasets is intolerable. With the advent of big data, these limitations become more and more urgent to solve.
- Second, they cannot adapt to sequential data or new coming data. In real life, data samples are usually collected sequentially as time passes, and some early collected data may be outdated. When the differences between the already collected data and the new coming data are large, current hashing methods usually lose their efficiency on the new data samples. Hence, it is important to develop online hash models, which can be efficiently updated to deal with sequential data.

In this paper, we overcome the limitations of batch mode hash methods [17]–[19], [21], [23]–[26] by developing an effective online hashing learning method called *Online Hashing* (OH). We propose a one-pass online adaptation criterion in a passive-aggressive way [39], which enables the newly learned hash model to embrace information from a new pair of data samples in the current round and meanwhile retain important information learned in the previous rounds. In addition, the exact labels of data are not required, and only the pairwise similarity is needed. More specifically, a similarity loss function is first designed to measure the confidence of the similarity between two hash codes of a pair of data samples, and then based on that similarity loss function a prediction loss function is proposed to evaluate whether the current hash model fits the current data under a structured prediction framework. We then minimize the proposed prediction loss function on the current input pair of data samples to update

Manuscript received November 27, 2015; revised July 14, 2016, November 10, 2016 and March 1, 2017; accepted March 24, 2017. This work was supported in part by National Key Research and Development Program of China(2016YFB1001002, 2016YFB1001003), by National Natural Science Foundation of China under Grant 61522115, Grant 61472456, Grant 61661130157, in part by Guangdong Natural Science Funds for Distinguished Young Scholar under Grant S2013050014265, by the Guangdong Program for Support of Top-notch Young Professionals under Grant 2014T Q01X779, and the RS-Newton Ad- vanced Fellowship (NA150459). The associate editor coordinating the review of this manuscript and approving it for publication was XXX. (Corresponding author: Wei- Shi Zheng)

L.-K. Huang is is with School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China (email: hlongkai@gmail.com).

Q. Yang is with School of Data and Computer Science, Sun Yat-sen University, Guangzhou, 510006, China, and with the Collaborative Innovation Center of High Performance Computing, National University of Defense Technology, Changsha 410073, China, (email: mmmyqmmm@gmail.com).

W.-S. Zheng is with School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China and also with the Key Laboratory of Machine Intelligence and Advanced Computing, Sun Yat-sen University, Ministry of Education, Guangzhou, China (e-mail: wszheng@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier

the hash model. During the online update, we wish to make the updated hash model approximate the model learned in the last round as much as possible for retaining the most historical discriminant information during the update. An upper bound on the cumulative similarity loss of the proposed online algorithm are derived, so that the performance of our online hash function learning can be guaranteed.

Since one-pass online learning only relies on the new data at the current round, the adaptation could be easily biased by the current round data. Hence, we introduce a multi-model online strategy in order to alleviate such a kind of bias, where multiple but not just one online hashing models are learned and they are expected to suit more diverse data pairs and will be selectively updated. A theoretical bound on the cumulative similarity loss is also provided.

In summary, the contributions of this work are

1) Developing a weakly supervised online hash function learning model. In our development, a novel similarity loss function is proposed to measure the difference of the hash codes of a pair of data samples in Hamming space. Following the similarity loss function is the prediction loss function to penalize the violation of the given similarity between the hash codes in Hamming space. Detailed theoretical analysis is presented to give a theoretical upper loss bound for the proposed online hashing method;
2) Developing a Multi-Model Online Hashing (MMOH), in which a multi-model similarity loss function is proposed to guide the training of multiple complementary hash models.

The rest of the paper is organized as follows. In Sec. II, related literatures are reviewed. In Sec. III, we present our online algorithm framework including the optimization method. Sec. IV further elaborates one left issue in Sec. III for acquiring zero-loss binary codes. Then we give analysis on the upper bound and time complexity of OH in Sec. V, and extend our algorithm to a multi-model one in Sec. VI. Experimental results for evaluation are reported in Sec. VII and finally we conclude the work in Sec. VIII.

II. Related Work

Online learning, especially one-pass online learning, plays an important role for processing large-scale datasets, as it is time and space efficient. It is able to learn a model based on streaming data, making dynamic update possible. In typical one-pass online learning algorithms [39], [40], when an instance is received, the algorithm makes a prediction, receives the feedback, and then updates the model based on this new data sample only upon the feedback. Generally, the performance of an online algorithm is guaranteed by the upper loss bound in the worst case.

There are a lot of existing works of online algorithms to solve specific machine learning problems [39]–[43]. However, it is difficult to apply these online methods to online hash function learning, because the sign function used in hash models is non-differentiable, which makes the optimization problem more difficult to solve. Although one can replace the sign function with sigmoid type functions or other approximate differentiable functions and then apply gradient descent, this becomes an obstacle on deriving the loss bound. There are existing works considering active learning and online learning together [44]–[46], but they are not for hash function learning.

Although it is challenging to design hash models in an online learning mode, several hashing methods are related to online learning [22], [41], [47]–[50]. In [41], the authors realized an online LSH by applying an online metric learning algorithm, namely LogDet Exact Gradient Online (LEGO) to LSH. Since [41] is operated on LSH, which is a data independent hash model, it does not directly optimize hash functions for generating compact binary code in an online way. The other five related works can be categorized into two groups: one is the stochastic gradient descent (SGD) based online methods, including minimal loss hashing [22], online supervised hashing [49] and Adaptive Hashing (AdaptHash) [50]; another group is matrix sketch based methods, including online sketching hashing (OSH) [47] and stream spectral binary coding (SSBC) [48].

Minimal loss hashing(MLH) [22] follows the loss-adjusted inference used in structured SVMs and deduces the convex-concave upper bound on the loss. Since MLH is a hash model relying on stochastic gradient decent update for optimization, it can naturally be used for online update. However, there are several limitations that make MLH unsuitable for online processing. First, the upper loss bound derived by MLH is actually related to the number of the historical samples used from the beginning. In other words, the upper loss bound of MLH may grow as the number of samples increases and therefore its online performance could not be guaranteed. Second, MLH assumes that all input data are centered (i.e. with zero mean), but such a pre-processing is challenging for online learning since all data samples are not available in advance.

Online supervised hashing (OECC) [49] is a SGD version of the Supervised Hashing with Error Correcting Codes (ECC) algorithm [51]. It employs a 0-1 loss function which outputs either 1 or 0 to indicate whether the binary code generated by existing hash model is in the codebook generated by error correcting output codes algorithm. If it is not in the codebook, the loss is 1. After replacing the 0-1 loss with a convex loss function and dropping the non-differentiable sign function in the hash function, SGD is applied to minimize the loss and update the hash model online. AdaptHash [50] is also a SGD based methods. It defines a loss function the same as the hingle-like loss function used in [22], [52]. To minimize this loss, the authors approximated the hash function by a differentiable sigmoid function and then used SGD to optimize the problem in an online mode. Both OECC and AdaptHash do not assume that data samples have zero mean as used in MLH. They handle the zero-mean issue by a method similar to the one in [52]. All these three SGD-based hashing methods enable online update by applying SGD, but they all cannot guarantee a constant loss upper bound.

Online sketching hashing (OSH) [47] was recently proposed to enable learning hash model on stream data by combining PCA hashing [27] and matrix sketching [53]. It first sketches stream samples into a small size matrix and meanwhile guar-

antee approximating data covariance, and then PCA hashing can be applied on this sketch matrix to learn a hash model. Sketching overcomes the challenge of training a PCA-based model on sequential data using limited storage. Stream spectral binary coding (SSBC) [48] is another learning to hash method on stream data based on the matrix sketch [53] skill. It applies matrix sketch processing on the Gaussian affinity matrix in spectral hashing algorithm [17]. Since the sketched matrix reserves global information of previously observed data, the new update model may not be adapted well on new observed data samples after a large number of samples have been sketched in the previous steps.

A preliminary version of this work was presented in [52]. Apart from more detailed analysis, this submission differs from the conference version in the following aspects: 1) we have further modified the similarity loss function to evaluate the hash model and to guide the update of the hash model;

2) we have developed a multi-model strategy to train multiple models to improve online hashing; 3) We have modified the strategy of zero-loss codes inference, which suits the passive aggressive scheme better; 4) much more experiments have been conducted to demonstrate the effectiveness of our method.

III. Online Modelling for Hash Function Learning

Hash models aim to learn a model to map a given data vector $\mathbf{x} \in \mathbb{R}^d$ to an r-bit binary code vector $\mathbf{h} \in \{-1,1\}^r$. For r-bit linear projection hashing, the $k^{th}(1 \leq k \leq r)$ hash function is defined as

$$h_k(\mathbf{x}) = sgn(\mathbf{w}_k^T\mathbf{x} + b_k) = \begin{cases} 1, & if \quad \mathbf{w}_k^T\mathbf{x} + b_k \geq 0, \\ -1, & otherwise, \end{cases}$$

where $\mathbf{w}_k \in \mathbb{R}^d$ is a projection vector, b_k is a scalar threshold, and $h_k \in \{-1,1\}$ is the binary hash code.

Regarding b_k in the above equation, a useful guideline proposed in [17], [27], [54] is that the scalar threshold b_k should be the median of $\{\mathbf{w}_k^T\mathbf{x_i}\}_{i=1}^n$, where n is the number of the whole input samples, in order to guarantee that half of the output codes are 1 while the other half are -1. This guarantees achieving maximal information entropy of every hash bit [17], [27], [54]. A relaxation is to set b_k to the mean of $\{\mathbf{w}_k^T\mathbf{x_i}\}_{i=1}^n$ and b_k will be zero if the data are zero-centered. Such a pretreatment not only helps to improve performance but also simplifies the hash function learning. Since data come in sequence, it is impossible to obtain the mean in advance. In our online algorithm, we will estimate the mean after a few data samples are available, update it after new data samples arrive, and perform update of zero-centered operation afterwards. Hence, the r-bit hash function becomes

$$\mathbf{h} = h(\mathbf{x}) = sgn(\mathbf{W}^T\mathbf{x}), \tag{1}$$

where $\mathbf{W} = [\mathbf{w_1}, \mathbf{w_2}, ..., \mathbf{w_r}] \in \mathbb{R}^{d\times r}$ is the hash projection matrix and $\mathbf{h} = [h_1(\mathbf{x}), h_2(\mathbf{x}), ..., h_r(\mathbf{x})]^T$ is the r-bit hash code. Here, $\mathbf{x}$ is the data point after zero-mean shifting.

Unfortunately, due to the non-differentiability of the sign function in Eq. (1), the optimization of the hash model becomes difficult. To settle such a problem, by borrowing the ideas from the structured prediction in structured SVMs [55], [56] and MLH [22], we transform Eq. (1) equivalently to the following structured prediction form:

$$\mathbf{h} = \underset{\mathbf{f}\in\{-1,1\}^r}{\arg\max}\ \mathbf{f}^T\mathbf{W}^T\mathbf{x}. \tag{2}$$

In the structured hash function Eq. (2), $\mathbf{f}^T\mathbf{W}^T\mathbf{x}$ can be regarded as a prediction value that measures the extent that the binary code $\mathbf{f}$ matches the hash code of $\mathbf{x}$ obtained through Eq. (1). Obviously, $\mathbf{f}^T\mathbf{W}^T\mathbf{x}$ is maximized only when each element of $\mathbf{f}$ has the same sign as that of $\mathbf{W}^T\mathbf{x}$.

A. Formulation

We start presenting the online hash function learning. In this work, we assume that the sequential data are in pairs. Suppose a new pair of data points $\mathbf{x_i}^t$ and $\mathbf{x_j}^t$ comes in the t^{th} $(t = 1, 2, ...)$ round with a similarity label s^t. The label indicates whether these two points are similar or not and it is defined as:

$$s^t = \begin{cases} 1, & if\ \mathbf{x_i}^t \text{ and } \mathbf{x_j}^t \text{ are similar}, \\ -1, & if\ \mathbf{x_i}^t \text{ and } \mathbf{x_j}^t \text{ are not similar}. \end{cases}$$

We denote the new coming pair of data points $\mathbf{x_i}^t$ and $\mathbf{x_j}^t$ by $\mathbb{x}^t = [\mathbf{x_i}^t, \mathbf{x_j}^t]$. In the t^{th} round, based on the hash projection matrix $\mathbf{W}^t$ learned in the last round, we can compute the hash codes of $\mathbf{x_i}^t$ and $\mathbf{x_j}^t$ denoted by $\mathbf{h_i}^t$, $\mathbf{h_j}^t$, respectively. Such a pair of hash codes is denoted by $\mathbb{h}^t = [\mathbf{h_i}^t, \mathbf{h_j}^t]$.

However, the prediction does not always match the given similarity label information of $\mathbb{x}^t$. When a mismatch case is observed, $\mathbf{W}^t$, the model learned in the $(t-1)^{th}$ round, needs to be updated for obtaining a better hash model $\mathbf{W}^{t+1}$. In this work, the Hamming distance is employed to measure the match or mismatch cases. In order to learn an optimal hash model that minimizes the loss caused by mismatch cases and maximizes the confidence of match cases, we first design a similarity loss function to quantify the difference between the pairwise hash codes $\mathbb{h}^t$ with respect to the corresponding similarity label s^t, which is formulated as follows:

$$R(\mathbb{h}^t, s^t) = \begin{cases} max\{0, \mathcal{D}_h(\mathbf{h_i}^t, \mathbf{h_j}^t) - \alpha\}, & if\ s^t = 1, \\ max\{0, \beta r - \mathcal{D}_h(\mathbf{h_i}^t, \mathbf{h_j}^t)\}, & if\ s^t = -1, \end{cases} \tag{3}$$

where $\mathcal{D}_h(\mathbf{h_i}^t, \mathbf{h_j}^t)$ is the Hamming distance between $\mathbf{h_i}^t$ and $\mathbf{h_j}^t$, α is an integer playing as the similarity threshold, and β is the dissimilarity ratio threshold ranging from 0 to 1. Generally, $\beta r > \alpha$, so that there is a certain margin between the match and mismatch cases.

In the above loss function, a relaxation is actually introduced by employing the threshold parameters α and βr as shown in Fig. 1. α should not be too large, so that the similarity of

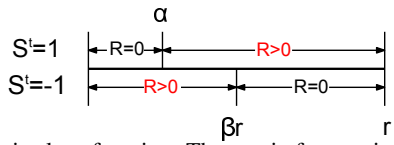


Fig. 1. Similarity loss function. The top is for a pair of similar data samples, and the bottom is for a pair of dissimilar ones.

the pairwise data samples can be preserved in the Hamming space. In contrast, βr should not be too small; otherwise two dissimilar data points cannot be well separated in Hamming space. From Fig. 1, we can see that the mismatch can be one of the following two cases: 1) the Hamming distance between the prediction codes $\mathbf{h_i}^t$ and $\mathbf{h_j}^t$ is larger than α for a similar pair; and 2) the Hamming distance between the prediction codes $\mathbf{h_i}^t$ and $\mathbf{h_j}^t$ is smaller than βr for a dissimilar pair. These two measure the risk of utilizing the already learned hash projection matrix $\mathbf{W}^t$ on a new pair of data points, i.e. $R(\mathbb{h}^t, s^t)$.

If the model learned in the last round predicts zero-loss hash code pair on a new pair, that is $R(\mathbb{h}^t, s^t) = 0$, our strategy is to retain the current model. If the model is unsatisfactory, i.e., predicting inaccurate hash codes having $R(\mathbb{h}^t, s^t) > 0$, we need to update the inaccurate previous hash projection matrix $\mathbf{W}^t$.

To update the hash model properly, we claim that the zero-loss hash code pair $\mathbb{g}^t = [\mathbf{g_i}^t, \mathbf{g_j}^t]$ for current round data pair $\mathbb{x}^t$ satisfying $R(\mathbb{g}^t, s^t) = 0$ is available, and we use the zero-loss hash code pair to guide update of the hash model, deriving the updated $\mathbf{W}^{t+1}$ towards a better prediction. We leave the details about how to obtain the zero-loss hash code pair presented in Sec. IV.

Now, we wish to obtain an updated hash projection matrix $\mathbf{W}^{t+1}$ such that it predicts similar hash code pair towards the zero-loss hash code pair $\mathbb{g}^t$ for the current input pair of data samples. Let us define

$$H^t(\mathbf{W}) = {\mathbf{h_i}^t}^T \mathbf{W}^T {\mathbf{x_i}}^t + {\mathbf{h_j}^t}^T \mathbf{W}^T {\mathbf{x_j}}^t, \tag{4}$$

$$G^t(\mathbf{W}) = {\mathbf{g_i}^t}^T \mathbf{W}^T {\mathbf{x_i}}^t + {\mathbf{g_j}^t}^T \mathbf{W}^T {\mathbf{x_j}}^t. \tag{5}$$

Given hash function Eq. (2) with respect to $\mathbf{W}^t$, we have $H^t(\mathbf{W}^t) \geq G^t(\mathbf{W}^t)$, since $\mathbf{h_i}^t$ and $\mathbf{h_j}^t$ are the binary solutions for $\mathbf{x_i}^t$ and $\mathbf{x_j}^t$ for the maximization, respectively, while $\mathbf{g_i}^t$ and $\mathbf{g_j}^t$ are not. This also suggests the $\mathbf{W}^t$ is not suitable for the generated binary code to approach the zero-loss hash code pair $\mathbb{g}^t$, and thus a new projection $\mathbf{W}^{t+1}$ has to be learned.

When updating the projection matrix from $\mathbf{W}^t$ to $\mathbf{W}^{t+1}$, we expect that the binary code generated for $\mathbf{x_i}^t$ is $\mathbf{g_i}^t$. According to the hash function in structured prediction form in Eq. (2), our expectation is to require ${\mathbf{g_i}^t}^T \mathbf{W}^{t+1^T} {\mathbf{x_i}}^t > {\mathbf{h_i}^t}^T \mathbf{W}^{t+1^T} {\mathbf{x_i}}^t$. Similarly, we expect the binary code generated for $\mathbf{x_j}^t$ is $\mathbf{g_j}^t$ and this is also to require ${\mathbf{g_j}^t}^T \mathbf{W}^{t+1^T} {\mathbf{x_j}}^t > {\mathbf{h_j}^t}^T \mathbf{W}^{t+1^T} {\mathbf{x_j}}^t$. Combining these two inequalities together, it would be expected that the new $\mathbf{W}^{t+1}$ should meet the condition that $G^t(\mathbf{W}^{t+1}) > H^t(\mathbf{W}^{t+1})$. To achieve this objective, we derive the following prediction loss function $\ell^t(\mathbf{W})$ for our algorithm:

$$\ell^t(\mathbf{W}) = H^t(\mathbf{W}) - G^t(\mathbf{W}) + \sqrt{R(\mathbb{h}^t, s^t)}. \tag{6}$$

In the above loss function, $\mathbb{h}^t$, $\mathbb{g}^t$ and $R(\mathbb{h}^t, s^t)$ are constants rather than variables dependent on $\mathbf{W}^t$. $R(\mathbb{h}^t, s^t)$ can be treated a loss penalization. When used in the Criterion (7) later, a small $R(\mathbb{h}^t, s^t)$ means a slight update is expected, and a large $R(\mathbb{h}^t, s^t)$ means a large update is necessary. Note that the square root of similarity loss function $R(\mathbb{h}^t, s^t)$ is utilized here, because it enables an upper bound on the cumulative loss functions, which will be shown in Sec.V-A.

Note that if $\ell^t(\mathbf{W}^{t+1}) = 0$, we can have $G^t(\mathbf{W}^{t+1}) = H^t(\mathbf{W}^{t+1}) + \sqrt{R(\mathbb{h}^t, s^t)} > H^t(\mathbf{W}^{t+1})$. Let $\widehat{\mathbb{g}}^t$ be the hash codes of $\mathbb{x}^t$ computed using the updated $\mathbf{W}^{t+1}$ by Eq. (2). Even though $G^t(\mathbf{W}^{t+1}) > H^t(\mathbf{W}^{t+1})$ cannot guarantee that $\widehat{\mathbb{g}}^t$ is exactly $\mathbb{g}^t$, it is probable that $\widehat{\mathbb{g}}^t$ is very close to $\mathbb{g}^t$ rather than $\mathbb{h}^t$. It therefore makes sense to force $\ell^t(\mathbf{W}^{t+1})$ to be zero or close to zero.

Since we are formulating a one-pass learning algorithm, the previously observed data points are not available for the learning in the current round, and the only information we can make use of is the current round projection matrix $\mathbf{W}^t$. In this case, we force that the newly learned $\mathbf{W}^{t+1}$ should stay close to the projection matrix $\mathbf{W}^t$ as much as possible so as to preserve the information learned in the last round as much as possible. Hence, the objective function for updating the hash projection matrix becomes

$$\begin{aligned} \mathbf{W}^{t+1} &= \arg\min_{\mathbf{W}} \frac{1}{2}||\mathbf{W} - \mathbf{W}^t||_F^2 + C\xi, \\ s.t. \quad & \ell^t(\mathbf{W}) \leq \xi \quad \text{and} \quad \xi \geq 0, \end{aligned} \tag{7}$$

where $\|\cdot\|_F$ is the Frobenius norm, ξ is a non-negative auxiliary variable to relax the constraint on the prediction loss function $\ell^t(\mathbf{W}) = 0$, and C is a margin parameter to control the effect of the slack term, whose influence will be observed in Sec.VII. Through this objective function, the difference between the new projection matrix $\mathbf{W}^{t+1}$ and the last one $\mathbf{W}^t$ is minimized, and meanwhile the prediction loss function $\ell^t(\mathbf{W})$ of the new $\mathbf{W}^{t+1}$ is bounded by a small value. We call the above model the *online hashing* (OH) model.

Finally, we wish to provide a comment on the function $H^t(\mathbf{W})$ in Eq. (4) and Eq. (6). Actually, an optimal case should be to refine function $H^t(\mathbf{W})$ as a function of variables $\mathbf{W}$ and a code pair $\mathbb{f} = [\mathbf{f_i}, \mathbf{f_j}] \in \{-1, 1\}^{r \times 2}$ as follows:

$$H^t(\mathbf{W}, \mathbb{f}) = {\mathbf{f_i}^t}^T \mathbf{W}^T {\mathbf{x_i}}^t + {\mathbf{f_j}^t}^T \mathbf{W}^T {\mathbf{x_j}}^t, \tag{8}$$

and then to refine the prediction loss function when an optimal update $\mathbf{W}^{t+1}$ is used:

$$\ell^t(\mathbf{W}^{t+1}) = \max_{\mathbb{f} \in \{-1,1\}^{r \times 2}} H^t(\mathbf{W}^{t+1}, \mathbb{f}) - G^t(\mathbf{W}^{t+1}) + \sqrt{R(\mathbb{h}^t, s^t)}. \tag{9}$$

The above refinement in theory can make $\max_{\mathbb{f}} H^t(\mathbf{W}^{t+1}, \mathbb{f}) - G^t(\mathbf{W}^{t+1})$ be a more rigorous loss on approximating the zero-loss hash code pair $\mathbb{g}^t$. But, it would be an obstacle to the optimization, since $\mathbf{W}^{t+1}$ is unknown when $\max_{\mathbb{f}} H^t(\mathbf{W}^{t+1}, \mathbb{f})$ is computed. Hence, we avert this problem by implicitly introducing an alternating optimization by first fixing $\mathbb{f}$ to be $\mathbb{h}^t$, then optimizing $\mathbf{W}^{t+1}$ by Criterion (7), and finally predicting the best $\mathbb{f}$ for $\max_{\mathbb{f} \in \{-1,1\}^{r \times 2}} H^t(\mathbf{W}^{t+1}, \mathbb{f})$. This process can be iterative. Although this may be useful to further improve our online model, we do not completely follow this implicit alternating processing to learn the $\mathbf{W}^{t+1}$ iteratively. This is because data are coming in sequence and it would be demanded to process a new data pair after an update of the projection matrix $\mathbf{W}$. Hence, in our implementation, we only update $\mathbf{W}^{t+1}$ once, and we provide the bound for $R(\mathbb{h}^t, s^t)$ under such a processing in Theorem 2.

B. Optimization

When $R(\mathbb{h}^t, s^t) = 0$, $\mathbb{h}^t$ is the optimal code pair and $\mathbb{g}^t$ is the same as $\mathbb{h}^t$, and thus $\ell^t(\mathbf{W}^t) = 0$. In this case, the solution to Criterion (7) is $\mathbf{W}^{t+1} = \mathbf{W}^t$. That is, when the already learned hash projection matrix $\mathbf{W}^t$ can correctly predict the similarity label of the new coming pair of data points $\mathbb{x}^t$, there is no need to update the hash function. When $R(\mathbb{h}^t, s^t) > 0$, the solution is

$$\begin{aligned} \mathbf{W}^{t+1} &= \mathbf{W}^t + \tau^t \mathbb{x}^t (\mathbb{g}^t - \mathbb{h}^t)^T, \\ \tau^t &= min\{C, \frac{\ell^t(\mathbf{W}^t)}{||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2}\}. \end{aligned} \tag{10}$$

The procedure for deriving the solution formulation (Eq. (10)) when $R(\mathbb{h}^t, s^t) > 0$ is detailed as follows.

First, the objective function (Criterion (7)) can be rewritten below when we introduce the Lagrange multipliers:

$$\mathcal{L}(\mathbf{W}, \tau^t, \xi, \lambda) = \frac{||\mathbf{W} - \mathbf{W}^t||_F^2}{2} + C\xi + \tau^t(\ell^t(\mathbf{W}) - \xi) - \lambda\xi, \tag{11}$$

where $\tau^t \geq 0$ and $\lambda \geq 0$ are Lagrange multipliers. Then, by computing $\partial\mathcal{L}/\partial\mathbf{W} = 0$, $\partial\mathcal{L}/\partial\xi = 0$, and $\partial\mathcal{L}/\partial\tau^t = 0$, we can have

$$\begin{aligned} 0 &= \frac{\partial\mathcal{L}}{\partial\mathbf{W}}, \\ \Rightarrow \mathbf{W} &= \mathbf{W}^t + \tau^t(\mathbf{x_i}^t(\mathbf{g_i}^t - \mathbf{h_i}^t)^T + \mathbf{x_j}^t(\mathbf{g_j}^t - \mathbf{h_j}^t)^T) \\ &= \mathbf{W}^t + \tau^t \mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T, \end{aligned} \tag{12}$$

$$\begin{aligned} 0 = \frac{\partial\mathcal{L}}{\partial\xi} &= C - \tau^t - \lambda, \\ \Rightarrow \ \tau^t &= C - \lambda. \end{aligned} \tag{13}$$

Since $\lambda > 0$, we have $\tau^t < C$. By putting Eq. (12) and Eq. (13) back into Eq. (6), we obtain

$$\ell^t(\mathbf{W}) = -\tau^t||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2 + \ell^t(\mathbf{W}^t). \tag{14}$$

Also, by putting Eqs. (12), (13) and (14) back into Eq. (11), we have

$$\mathcal{L}(\tau^t) = -\frac{1}{2}\ {\tau^t}^2||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2 + \tau^t\ell^t(\mathbf{W}^t).$$

By taking the derivative of $\mathcal{L}$ with respect to τ^t and setting it to zero, we get

$$\begin{aligned} 0 = \frac{\partial\mathcal{L}}{\partial\tau^t} &= -\tau^t||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2 + \ell^t(\mathbf{W}^t), \\ \Rightarrow \quad \tau^t &= \frac{\ell^t(\mathbf{W}^t)}{||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2}. \end{aligned} \tag{15}$$

Since $\tau^t < C$, we can obtain

$$\tau^t = min\{C,\ \frac{\ell^t(\mathbf{W}^t)}{||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2}\}. \tag{16}$$

In summary, the solution to the optimization problem in Criterion (7) is Eq. (10), and the whole procedure of the proposed OH is presented in Algorithm 1.

Algorithm 1 Online Hashing

INITIALIZE $\mathbf{W}^1$
for t = 1,2,... **do**
  Receive a pairwise instance $\mathbb{x}^t$ and similarity label s^t;
  Compute the hash code pair $\mathbb{h}^t$ of $\mathbb{x}^t$ by Eq. (1);
  Compute the similarity loss $R(\mathbb{h}^t, s^t)$ by Eq. (3);
  if $R(\mathbb{h}^t, s^t) > 0$ **then**
    Get the zero-loss code pair $\mathbb{g}^t$ that makes $R(\mathbb{g}^t, s^t) = 0$;
    Compute the prediction loss $\ell^t(\mathbf{W}^t)$ by Eq. (6);
    Set $\tau^t = min\{C, \frac{\ell^t(\mathbf{W}^t)}{||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2}\}$;
    Update $\mathbf{W}^{t+1} = \mathbf{W}^t + \tau^t\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T$;
  else
    $\mathbf{W}^{t+1} = \mathbf{W}^t$;
  end if
end for

C. Kernelization

Kernel trick is well-known to make machine learning models better adapted to nonlinearly separable data [21]. In this context, a kernel-based OH is generated by employing explicit kernel mapping to cope with the nonlinear modeling. In details, we aim at mapping data in the original space $\mathbb{R}^d$ into a feature space $\mathbb{R}^m$ through a kernel function based on m ($m < d$) anchor points, and therefore we have a new representation of $\mathbf{x}$ which can be formulated as follows:

$$z(\mathbf{x}) = [\kappa(\mathbf{x}, \mathbf{x}_{(1)}), \kappa(\mathbf{x}, \mathbf{x}_{(2)}), ..., \kappa(\mathbf{x}, \mathbf{x}_{(m)})]^T,$$

where $\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, ..., \mathbf{x}_{(m)}$ are m anchors.

For our online hash model learning, we assume that at least m data points have been provided in the initial stage; otherwise, the online learning will not start until at least m data points have been collected, and then these m data points are considered as the m anchors used in the kernel trick. Regarding the kernel used in this work, we employ the Gaussian RBF kernel $\kappa(\mathbf{x}, \mathbf{y}) = exp(-||\mathbf{x} - \mathbf{y}||^2/2\sigma^2)$, where we set σ to $\mathbf{1}$ in our algorithm.

IV. Zero-loss Binary Code Pair Inference

In Sec.III-A, we have mentioned that our online hashing algorithm relies on the zero-loss code pair $\mathbb{g}^t = [\mathbf{g_i}^t, \mathbf{g_j}^t]$ which satisfies $R(\mathbb{g}^t, s^t) = 0$. Now, we detail how to acquire $\mathbb{g}^t$.

Dissimilar Case. We first present the case for dissimilar pairs. As mentioned in Sec. III-A, to achieve zero similarity loss, the Hamming distance between the hash codes of non-neighbors should not be smaller than βr. Therefore, we need to seek the $\mathbb{g}^t$ such that $D_h(\mathbf{g_i}^t, \mathbf{g_j}^t) \geq \beta r$. Denote the k^{th} bit of $\mathbf{h_i}^t$ by $\mathbf{h_i}^t_{[k]}$, and similarly we have $\mathbf{h_j}^t_{[k]}, \mathbf{g_i}^t_{[k]}, \mathbf{g_j}^t_{[k]}$. Then $D_h(\mathbf{h_i}^t, \mathbf{h_j}^t) = \sum_{k=1}^r D_h(\mathbf{h_i}^t_{[k]}, \mathbf{h_j}^t_{[k]})$, where

$$D_h(\mathbf{h_i}^t_{[k]}, \mathbf{h_j}^t_{[k]}) = \begin{cases} 0, & if \quad \mathbf{h_i}^t_{[k]} = \mathbf{h_j}^t_{[k]}, \\ 1, & if \quad \mathbf{h_i}^t_{[k]} \neq \mathbf{h_j}^t_{[k]}. \end{cases}$$

Let $\mathcal{K}_1 = \{k|D_h(\mathbf{h_i}^t_{[k]}, \mathbf{h_j}^t_{[k]}) = 1\}$ and $\mathcal{K}_0 = \{k|D_h(\mathbf{h_i}^t_{[k]}, \mathbf{h_j}^t_{[k]}) = 0\}$. To obtain $\mathbb{g}^t$, we first set $\mathbf{g_i}^t_{[k]} = \mathbf{h_i}^t_{[k]}$ and $\mathbf{g_j}^t_{[k]} = \mathbf{h_j}^t_{[k]}$ for $k \in \mathcal{K}_1$, so as to retain the Hamming distance obtained through the hash model learned in the last round. Next, in order to increase the Hamming distance, we need to make $D_h(\mathbf{g_i}^t_{[k]}, \mathbf{g_j}^t_{[k]}) = 1$ for the $k \in \mathcal{K}_0$. That is, we

need to set[1] either $\mathbf{g_i}^t_{[k]} = -\mathbf{h_i}^t_{[k]}$ or $\mathbf{g_j}^t_{[k]} = -\mathbf{h_j}^t_{[k]}$, for all the $k \in \mathcal{K}_0$. Hence, we can pick up p bits whose indexes are in set $\mathcal{K}_0$ to change/update such that

$$D_h(\mathbf{g_i}^t, \mathbf{g_j}^t) = D_h(\mathbf{h_i}^t, \mathbf{h_j}^t) + p. \tag{17}$$

Now the problem is how to set p, namely the number of hash bits to update. We first investigate the relationship between the update of projection vectors and $\mathbb{g}^t$. Note that $\mathbf{W}$ consists of r projection vectors $\mathbf{w}_k$ $(k = 1, 2, ..., r)$. From Eq. (12), we can deduce that

$$\mathbf{w_k^{t+1}} = \mathbf{w_k^t} + \tau^t(\mathbf{x_i}^t(\mathbf{g_i}^t_{[k]} - \mathbf{h_i}^t_{[k]}) + \mathbf{x_j}^t(\mathbf{g_j}^t_{[k]} - \mathbf{h_j}^t_{[k]})). \tag{18}$$

It can be found that $\mathbf{w_k^{t+1}} = \mathbf{w_k^t}$, when $\mathbf{g_i}^t_{[k]} = \mathbf{h_i}^t_{[k]}$ and $\mathbf{g_j}^t_{[k]} = \mathbf{h_j}^t_{[k]}$; otherwise, $\mathbf{w_k^t}$ will be updated. So the more $\mathbf{w}_k$ in $\mathbf{W}$ we update, the more corresponding hash bits of all data points we subsequently have to update when applied to real-world system. This takes severely much time which cannot be ignored for online applications. Hence, we should change hash bits as few as possible; in other words, we aim to update $\mathbf{w}_k$ as few as possible. This means that p should be as small as possible, meanwhile guaranteeing that $\mathbb{g}^t$ satisfies the constrain $R(\mathbb{g}^t, s^t) = 0$. Based on the above discussion, the minimum of p is computed as $p_0 = \lceil \beta r \rceil - D_h(\mathbf{h_i}^t, \mathbf{h_j}^t)$ by setting $D_h(\mathbf{g_i}^t, \mathbf{g_j}^t) = \lceil \beta r \rceil$, as $p = D_h(\mathbf{g_i}^t, \mathbf{g_j}^t) - D_h(\mathbf{h_i}^t, \mathbf{h_j}^t)$ and $D_h(\mathbf{g_i}^t, \mathbf{g_j}^t) \geq \lceil \beta r \rceil \geq \beta r$. Then $\mathbb{g}^t$ is ready by selecting p_0 hash bits whose indexes are in $\mathcal{K}_0$.

After determining the number of hash bits to update, namely p_0, the problem now becomes which p_0 bits should be picked up from $\mathcal{K}_0$. To establish the rule, it is necessary to measure the potential loss for every bit of $\mathbf{h}_i^t$ and $\mathbf{h}_j^t$. For this purpose, the prediction loss function in Eq. (6) can be reformed as

$$\sum_{\mathbf{h_i}^t_{[k]} \neq \mathbf{g_i}^t_{[k]}} 2\mathbf{h_i}^t_{[k]} \mathbf{w_k^t}^T \mathbf{x_i}^t + \sum_{\mathbf{h_j}^t_{[k]} \neq \mathbf{g_j}^t_{[k]}} 2\mathbf{h_j}^t_{[k]} \mathbf{w_k^t}^T \mathbf{x_j}^t + \sqrt{R(\mathbb{h}^t, s^t)}.$$

This tells that $\mathbf{h_i}^t_{[k]} \mathbf{w_k^t}^T \mathbf{x_i}^t$ or $\mathbf{h_j}^t_{[k]} \mathbf{w_k^t}^T \mathbf{x_j}^t$ are parts of the prediction loss, and thus we use it to measure the potential loss for every bit. The problem is which bit should be picked up to optimize. For our one-pass online learning, a large update does not mean a good performance will be gained since every time we update the model only based on a new arrived pair of samples, and thus a large change on the hash function would not suit the passed data samples very well. This also conforms to the spirit of passive-aggressive idea that the change of an online model should be smooth. To this end, we take a conservative strategy by selecting the p_0 bits that corresponding to smallest potential loss as introduced below. First, the potential loss of every bit w.r.t $H(\mathbf{W}^t)$ is calculated by

$$\delta_k = min\{\mathbf{h_i}^t_{[k]} \mathbf{w_k^t}^T \mathbf{x_i}^t, \quad \mathbf{h_j}^t_{[k]} \mathbf{w_k^t}^T \mathbf{x_j}^t\}, \qquad k \in \mathcal{K}_0. \tag{19}$$

We only select the smaller one between $\mathbf{h_i}^t_{[k]} \mathbf{w_k^t}^T \mathbf{x_i}^t$ and $\mathbf{h_j}^t_{[k]} \mathbf{w_k^t}^T \mathbf{x_j}^t$ because we will never set $\mathbb{g}^t$ simultaneously by $\mathbf{g_i}^t_{[k]} = -\mathbf{h_i}^t_{[k]}$ and $\mathbf{g_j}^t_{[k]} = -\mathbf{h_j}^t_{[k]}$ for any $k \in \mathcal{K}_0$. After sorting δ_k, the p_0 smallest δ_k are picked up and their corresponding hash bits are updated by the following rule:

$$\begin{cases} \mathbf{g_i}_{[k]} = -\mathbf{h_i}_{[k]}, \ if \ \mathbf{h_i}^t_{[k]} \mathbf{w_k^t}^T \mathbf{x_i}^t \leq \mathbf{h_j}^t_{[k]} \mathbf{w_k^t}^T \mathbf{x_j}^t, \\ \mathbf{g_j}_{[k]} = -\mathbf{h_j}_{[k]}, \ otherwise. \end{cases} \tag{20}$$

The procedure of obtaining $\mathbb{g}^t$ for a dissimilar pair is summarized in Algorithm 2.

Similar Case. Regarding similar pairs, the Hamming distance of the optimal hash code pairs $\mathbb{g}^t$ should be equal or smaller than α. Since the Hamming distance between the predicted hash codes of similar pairs may be larger than α, we should pick up p_0 bits from set $\mathcal{K}_1$ instead of from set $\mathcal{K}_0$, and set them opposite to the corresponding values in $\mathbb{h}^t$ so as to achieve $R(\mathbb{g}^t, s^t) = 0$. Similar to the case for dissimilar pairs as discussed above, the number of hash bits to be updated is $p_0 = D_h(\mathbf{h_i}^t, \mathbf{h_j}^t) - \alpha$, but these bits are selected in $\mathcal{K}_1$. We will compute δ_k for $k \in \mathcal{K}_1$ and pick up p_0 bits with the smallest δ_k for update. Since the whole processing is similar to the processing for the dissimilar pairs, we only summarize the processing for similar pairs in Algorithm 3 and skip the details.

Finally, when $\mathbf{w_k^t}$ is a zero vector, δ_k is zero as well no matter what the values of $\mathbf{h_i}^t_{[k]}$, $\mathbf{h_j}^t_{[k]}$, $\mathbf{x_i}^t$ and $\mathbf{x_j}^t$ are. This leads to the failure in selecting hash bits to be updated. To avert this, we initialize $\mathbf{W}^1$ by applying LSH. In other words, $\mathbf{W}^1$ is sampled from a zero-mean multivariate Gaussian $\mathcal{N}(0, I)$, and we denote this matrix by $\mathbf{W}_{LSH}$.

Algorithm 2 Inference of $\mathbb{g}^t$ for a dissimilar pair

Calculate the Hamming distance $D_h(\mathbf{h_i}^t, \mathbf{h_j}^t)$ between $\mathbf{h_i}^t$ and $\mathbf{h_j}^t$;
Calculate $p_0 = \lceil \beta r \rceil - D_h(\mathbf{h_i}^t, \mathbf{h_j}^t)$;
Compute δ_k for $k \in \mathcal{K}_0$ by Eq. (19);
Sort δ_k;
Set the corresponding hash bits of the p_0 smallest δ_k opposite to the corresponding ones in $\mathbb{h}^t$ by following the rule in Eq.(20) and keep the others in $\mathbb{h}^t$ without change.

Algorithm 3 Inference of $\mathbb{g}^t$ for a similar pair

Calculate the Hamming distance $D_h(\mathbf{h_i}^t, \mathbf{h_j}^t)$ between $\mathbf{h_i}^t$ and $\mathbf{h_j}^t$;
Calculate $p_0 = D_h(\mathbf{h_i}^t, \mathbf{h_j}^t) - \alpha$;
Compute δ_k for $k \in \mathcal{K}_1$ by Eq. (19);
Sort δ_k;
Set the corresponding hash bits of the p_0 smallest δ_k opposite to the corresponding values in $\mathbb{h}^t$ by following the rule in Eq.(20) and keep the others in $\mathbb{h}^t$ with no change.

V. Analysis

A. Bounds for Similarity Loss and Prediction Loss

In this section, we discuss the loss bounds for the proposed online hashing algorithm. For convenience, at step t, we define

$$\ell_U^t = \ell^t(\mathbf{U}) = H^t(\mathbf{U}) - G^t(\mathbf{U}) + \sqrt{R(\mathbb{h}^t, s^t)}, \tag{21}$$

where $\mathbf{U}$ is an arbitrary matrix in $\mathbb{R}^{d \times r}$. Here, ℓ_U^t is considered as the prediction loss based on $\mathbf{U}$ in the t^{th} round.

We first present a lemma that will be utilized to prove Theorem 2.

[1]The hash code in our algorithm is -1 and 1. Note that, $\mathbf{g_i}^t_{[k]} = -\mathbf{h_i}^t_{[k]}$ means set $\mathbf{g_i}^t_{[k]}$ to be different from $\mathbf{h_i}^t_{[k]}$

Lemma 1. *Let $(\mathbb{x}^1, s^1), \cdots, (\mathbb{x}^t, s^t)$ be a sequence of pairwise examples, each with a similarity label $s^t \in \{1, -1\}$. The data pair $\mathbb{x}^t \in \mathbb{R}^{d\times 2}$ is mapped to a r-bit hash code pair $\mathbb{h}^t \in \mathbb{R}^{r\times 2}$ through the hash projection matrix $\mathbf{W}^t \in \mathbb{R}^{d\times r}$. Let $\mathbf{U}$ be an arbitrary matrix in $\mathbb{R}^{d\times r}$. If τ^t is defined as that in Eq. (10), we then have*

$$\sum_{t=1}^{\infty} \tau^t (2\ell^t(\mathbf{W}^t) - \tau^t ||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2 - 2\ell_U^t) \leq ||\mathbf{U} - \mathbf{W}^1||_F^2,$$

where $\mathbf{W}^1$ is the initialized hash projection matrix that consists of non-zero vectors.

Proof. By using the definition

$$\Delta_t = ||\mathbf{W}^t - \mathbf{U}||_F^2 - ||\mathbf{W}^{t+1} - \mathbf{U}||_F^2,$$

we can have

$$\begin{aligned} \sum_{t=1}^{\infty} \Delta_t &= \sum_{t=1}^{\infty} (||\mathbf{W}^t - \mathbf{U}||_F^2 - ||\mathbf{W}^{t+1} - \mathbf{U}||_F^2) \\ &= ||\mathbf{W}^1 - \mathbf{U}||_F^2 - ||\mathbf{W}^{t+1} - \mathbf{U}||_F^2 \leq ||\mathbf{W}^1 - \mathbf{U}||_F^2. \end{aligned} \tag{22}$$

From Eq. (12), we know $\mathbf{W}^{t+1} = \mathbf{W}^t + \tau^t \mathbb{x}^t (\mathbb{g}^t - \mathbb{h}^t)^T$, so we can rewrite Δ_t as

$$\begin{aligned} \Delta_t &= ||\mathbf{W}^t - \mathbf{U}||_F^2 - ||\mathbf{W}^{t+1} - \mathbf{U}||_F^2 \\ &= ||\mathbf{W}^t - \mathbf{U}||_F^2 - ||\mathbf{W}^t - \mathbf{U} + \tau^t \mathbb{x}^t (\mathbb{g}^t - \mathbb{h}^t)^T||_F^2 \\ &= ||\mathbf{W}^t - \mathbf{U}||_F^2 - (||\mathbf{W}^t - \mathbf{U}||_F^2 \\ &\quad + 2\tau^t (H^t(\mathbf{W}) - G^t(\mathbf{W}) - (H^t(\mathbf{U}) - G^t(\mathbf{U}))) \\ &\quad + (\tau^t)^2 ||\mathbb{x}^t (\mathbb{g}^t - \mathbb{h}^t)^T||_F^2) \\ &\geq -2\tau^t ((\sqrt{R(\mathbb{h}^t, s^t)} - \ell^t(\mathbf{W}^t)) - (\sqrt{R(\mathbb{h}^t, s^t)} - \ell_U^t)) \\ &\quad - (\tau^t)^2 ||\mathbb{x}^t (\mathbb{g}^t - \mathbb{h}^T)^T||_F^2 \\ &= \tau^t (2\ell^t(\mathbf{W}^t) - \tau^t ||(\mathbb{x}^t (\mathbb{g}^t - \mathbb{h}^t)^T||_F^2 - 2\ell_U^t). \end{aligned}$$

By computing the sum of the left and the right of the above inequality we can obtain

$$\sum_{t=1}^{\infty} \Delta_t \geq \sum_{t=1}^{\infty} \tau^t (2\ell^t(\mathbf{W}^t) - \tau^t ||(\mathbb{x}^t (\mathbb{g}^t - \mathbb{h}^t)^T||_F^2 - 2\ell_U^t).$$

Finally, putting Eq. (22) into the above equation, we prove Lemma 1. □

Theorem 2. *Let $(\mathbb{x}^1, s^1), \cdots, (\mathbb{x}^t, s^t)$ be a sequence of pairwise examples, each with a similarity label $s^t \in \{1, -1\}$ for all t. The data pair $\mathbb{x}^t \in \mathbb{R}^{d\times 2}$ is mapped to a r-bit hash code pair $\mathbb{h}^t \in \mathbb{R}^{r\times 2}$ through the hash projection matrix $\mathbf{W}^t \in \mathbb{R}^{d\times r}$. If $||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2$ is upper bounded by F^2 and the margin parameter C is set as the upper bound of $\frac{\sqrt{R(\mathbb{h}^t, s^t)}}{F^2}$, then the cumulative similarity loss (Eq. (3)) is bounded for any matrix $\mathbf{U} \in \mathbb{R}^{d\times r}$, i.e.*

$$\sum_{t=1}^{\infty} R(\mathbb{h}^t, s^t) \leq F^2 (||\mathbf{U} - \mathbf{W}^1||_F^2 + 2C \sum_{t=1}^{\infty} \ell_U^t),$$

where C is the margin parameter defined in Criterion (7).

Proof. Based on Lemma 1, we can obtain

$$\begin{aligned} &\sum_{t=1}^{\infty} \tau^t (2\ell^t(\mathbf{W}^t) - \tau^t ||(\mathbb{x}^t (\mathbb{g}^t - \mathbb{h}^t)^T||_F^2) \\ &\qquad \leq ||\mathbf{U} - \mathbf{W}^1||_F^2 + 2\sum_{t=1}^{\infty} \tau^t \ell_U^t. \end{aligned} \tag{23}$$

Based on Eq. (16), we get that

$$\frac{\ell^t(\mathbf{W}^t)}{||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2} \geq \tau^t.$$

This deduces that

$$\begin{aligned} &\tau^t (2\ell^t(\mathbf{W}^t) - \tau^t ||(\mathbb{x}^t (\mathbb{g}^t - \mathbb{h}^t)^T||_F^2) \\ \geq &\tau^t (2\ell^t(\mathbf{W}^t) - \ell^t(\mathbf{W}^t)) \;=\; \tau^t \ell^t(\mathbf{W}^t). \end{aligned} \tag{24}$$

According to the definition of prediction loss function in Eq. (6) and the upper bound assumption, we know that for any t,

$$\begin{aligned} &\sqrt{R(\mathbb{h}^t, s^t)} \leq \ell^t(\mathbf{W}^t), \\ &||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2 \leq F^2, \text{ and} \\ &\frac{\sqrt{R(\mathbb{h}^t, s^t)}}{F^2} \leq C \end{aligned} \tag{25}$$

With these three inequalities and Eq.(16) , it can be deduced that

$$\begin{aligned} \tau^t \ell^t(\mathbf{W}^t) &= min \left\{ \frac{\ell^t(\mathbf{W}^t)^2}{||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2}, C\ell^t(\mathbf{W}^t) \right\} \\ &\geq min \left\{ \frac{R(\mathbb{h}^t, s^t)}{||\mathbb{x}^t(\mathbb{g}^t - \mathbb{h}^t)^T||_F^2}, C\sqrt{R(\mathbb{h}^t, s^t)} \right\} \\ &\geq min \left\{ \frac{R(\mathbb{h}^t, s^t)}{F^2}, \frac{R(\mathbb{h}^t, s^t)}{F^2} \right\} \\ &= \frac{R(\mathbb{h}^t, s^t)}{F^2}. \end{aligned} \tag{26}$$

By combining Eq. (23) and Eq. (26), we obtain that

$$\sum_{t=1}^{\infty} \frac{R(\mathbb{h}^t, s^t)}{F^2} \leq ||\mathbf{U} - \mathbf{W}^1||_F^2 + 2\sum_{t=1}^{\infty} \tau^t \ell_U^t.$$

Since, $\tau^t \leq C$ for all t, we have

$$\sum_{t=1}^{\infty} R(\mathbb{h}^t, s^t) \leq F^2 (||\mathbf{U} - \mathbf{W}^1||_F^2 + 2C \sum_{t=1}^{\infty} \ell_U^t). \tag{27}$$

The theorem is proven. □

Note: In particular, if the assumption that there exists a projection matrix satisfying zero or negative prediction loss for any pair of data samples holds, there would exist a $\mathbf{U}$ satisfying $\ell_U^t \leq 0$ for any t. If so, the second term in inequality (27) vanishes. In other words, the similarity loss is bounded by a constant, i.e. $F^2||\mathbf{U} - \mathbf{W}^1||_F^2$. After a certain amount of update, the similarity loss for new data sample pairs becomes zero in such an optimistic case.

B. Time and Space Complexity

Based on the algorithm summarized in Algorithm 1, we can find that the time of computing the prediction code for $\mathbb{x}^t$ is $O(dr)$ and that of obtaining the similarity loss is $O(r)$. The process of obtaining the zero-loss code pair $\mathbb{g}^t$ takes at most $O(rlogr + rd)$ with $O(rd)$ to compute all δ_k and $O(rlogr)$ to sort hash bits according to δ_k. As for the update process of the projection matrix, it takes $O(rd)$. Therefore, the time complexity for training OH at each round is $O(dr + rlogr)$. Overall, if n pairs of data points participate in the training

stage, the whole time complexity is $O((dr + rlogr)n)$. For the space complexity, it is $O(d + dr) = O(dr)$, with $O(d)$ to store the data pairs and $O(dr)$ to store the projection matrix. Overall, the space complexity remains unchanged during training and is independent of the number of training samples.

VI. Multi-Model Online Hashing

In order to make the online hashing model more robust and less biased by current round update, we extend the proposed online hashing from updating one single model to updating the T models. Suppose that we are going to train T models, which are initialized randomly by LSH. Each model is associated to the optimization of its own similarity loss function in terms of Eq. (3), denoted by $R_m(\mathbb{h}_m^t, s^t)$ $(m = 1, 2, \cdots, T)$, where $\mathbb{h}_m^t$ is the binary code of a new pair $\mathbb{x}^t$ predicted by the m^{th} model at step t. At step t, if $\mathbb{x}^t$ is a similar pair, we only select one of the T models to update. To do that, we compute the similarity loss function for each model $R_m(\mathbb{h}_m^t, s^t)$, and then we select the model, supposed the m_0^{th} model that obtains the smallest similarity loss, i.e., $m_0 = \arg\min_m R_m(\mathbb{h}_m^t, s^t)$. Note that for a similar pair, it is enough that one of the models has positive output, and thus the selected model is the closest one to suit this similar pair and is more easier to update. If $\mathbb{x}^t$ is a dissimilar pair, all models will be updated if the corresponding loss is not zero, since we cannot tolerate an wrong prediction for a dissimilar pair. By performing online hashing in this way, we are able to learn diverse models that could fit different data samples locally. The update of each model follows the algorithm presented in our last section.

To guarantee the rationale of the multi-model online hashing, we also provide the upper bound for the accumulative multi-model similarity loss in the theorem below.

Theorem 3. *Let $(\mathbb{x}^1, s^1), \cdots, (\mathbb{x}^t, s^t)$ be a sequence of pairwise examples, each with a similarity label $s^t \in \{1, -1\}$ for all t. The data pair $\mathbb{x}^t \in \mathbb{R}^{d\times 2}$ is mapped to a r-bit hash code pair $\mathbb{h}^t \in \mathbb{R}^{r\times 2}$ through the hash projection matrix $\mathbf{W}^t \in \mathbb{R}^{d\times r}$. Suppose $||\mathbb{x}^t(\mathbb{g}_m^t - \mathbb{h}_\mathbb{m}^\mathbb{t})^T||_F^2$ is upper bounded by F^2, and the margin parameter C is set as the upper bound of $\frac{\sqrt{R_m^*(\mathbb{h}_m^t, s^t)}}{F^2}$ for all m, where $R_m^*(\mathbb{h}_m^t, s^t)$ is an auxiliary function defined as:*

$$R_m^*(\mathbb{h}_m^t, s^t) = \begin{cases} R_m(\mathbb{h}_m^t, s^t), & \text{if the } m^{th} \text{ model is selected for update at step } t, \\ 0, & otherwise. \end{cases} \tag{28}$$

Then for any matrix $\mathbf{U} \in \mathbb{R}^{d\times r}$, the cumulative similarity loss (Eq. (3)) is bounded, i.e.,

$$\sum_{t=1}^{\infty}\sum_{m=1}^{T} R_m^*(\mathbb{h}_m^t, s^t) \le TF^2(||\mathbf{U} - \mathbf{W}^1||_F^2 + 2C\sum_{t=1}^{\infty}\ell_U^t),$$

where C is the margin parameter defined in Criterion (7).

Proof. Based on Theorem 2, the following inequality holds for $m = 1, 2, ...T$:

$$\sum_{t=1}^{\infty} R_m^*(\mathbb{h}_m^t, s^t) \le F^2(||\mathbf{U} - \mathbf{W}^1||_F^2 + 2C\sum_{t=1}^{\infty}\ell_U^t).$$

By summing these multi-model similarity losses of all models, Theorem 3 is proved. □

VII. Experiments

In this section, extensive experiments were conducted to verify the efficiency and effectiveness of the proposed OH models from two aspects: metric distance neighbor search and semantic neighbor search. First, four selected datasets are introduced in Sec. VII-A. And then, we evaluate the proposed models in Sec. VII-B. Finally, we make comparison between the proposed algorithms and several related hashing models in Sec. VII-C.

A. Datasets

The four selected large-scale datasets are: Photo Tourism [57], 22K LabelMe [58], GIST1M [59] and CIFAR-10 [60], which are detailed below.

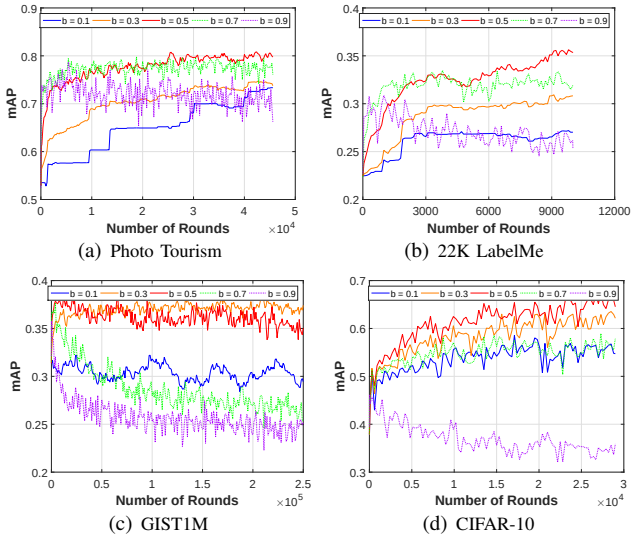
Photo Tourism [57]. It is a large collection of 3D photographs including three subsets, each of which has about 100K patches with 64×64 grayscale. In the experiment, we selected one subset consisting of 104K patches taken from Half Dome in Yosemite. We extracted 512-dimensional GIST feature vector for each patch and randomly partitioned the whole dataset into a training set with 98K patches and a testing set with 6K patches. The pairwise label s^t is generated based on the matching information. That is, s^t is 1 if a pair of patches is matched; otherwise s^t is -1.

22K LabelMe [58]. It contains 22,019 images. In the experiment, each image was represented by 512-dimensional GIST feature vector. We randomly selected $2K$ images from the dataset as the testing set, and set the remaining images as the training set. To set the similarity label between two data samples, we followed [21], [27]: if either one is within the top 5% nearest neighbors of the other measured by Euclidean distance, $s^t = 1$ (i.e., they are similar); otherwise $s^t = -1$ (i.e., they are dissimilar).

Gist1M [59]. It is a popular large-scale dataset to evaluate hash models [20], [61]. It contains one million unlabeled data with each data represented by a 960-dimensional GIST feature vector. In the experiment, we randomly picked up 500,000 points for training and the non-overlapped 1,000 points for testing. Owing to the absence of label information, we utilize pseudo label information by thresholding the top 5% of the whole dataset as the true neighbors of an instance based on Euclidean distance, so every point has 50,000 neighbors.

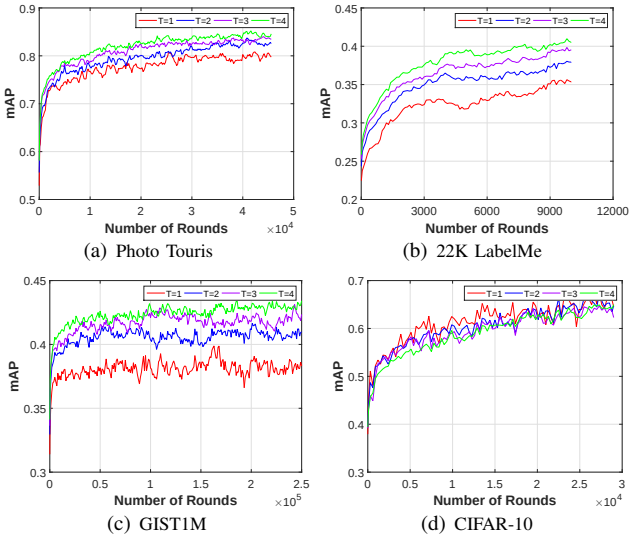
CIFAR-10 and Tiny Image 80M [60]. CIFAR-10 is a labeled subset of the 80M Tiny Images collection [62]. It consists of 10 classes with each class containing $6K$ 32×32 color images, leading to 60K images in total. In the experiment, every image was represented by 2048-dimensional deep features, and 59K samples were randomly selected to set up the training set with the remained 1K as queries to search through the whole 80M Tiny Image collection.

For measurement, the mean average precision (mAP) [63], [64] is used to measure the performance of different algorithms, and mAP is regarded as a better measure than precision and recall when evaluating the quality of results in retrieval [63], [64]. All experiments were independently run on a server with CPU Intel Xeon X5650, 12 GB memory and 64-bit CentOS system.



(a) Photo Tourism (b) 22K LabelMe (c) GIST1M (d) CIFAR-10

Fig. 2. mAP comparison results of OH with respect to different β on all datasets. (Best viewed in color.)



(a) Photo Touris (b) 22K LabelMe (c) GIST1M (d) CIFAR-10

Fig. 3. mAP comparison results of MMOH with respect to different T on all datasets. (Best viewed in color.)

B. Evaluation of the Proposed Methods

In the proposed models, there are three key parameters, namely β, C and T. In this subsection, we mainly investigate the influence of these parameters. Additionally, we will observe the influence of the RBF kernel function on the proposed models.

As stated in the previous section, the initial projection matrix $\mathbf{W}^1$ is randomly set based on Gaussian distribution, which is similar to the generation of $\mathbf{W}$ in LSH [6]. Thus, such an initialization of $\mathbf{W}^1$ is denoted by $\mathbf{W}^1 = \mathbf{W}_{LSH}$. For the similarity threshold, α is set to 0, because in most applications, the nearest neighbors of a given sample are looked up within 0 Hamming distance. The dissimilar ratio threshold, β, is set to 0.4 on CIFAR-10 and Gist 1M and set to 0.5 on the other two datasets. Besides, the code length r is set as $r = 64$ in this section, and the RBF kernel is a default kernel used in the proposed models in the experiments. When a parameter is being evaluated, the others are set as the default values as shown in Table I.

TABLE I
THE DEFAULT VALUES OF THE PARAMETERS IN MMOH

parameter	T	r	$\mathbf{W}^1$	α	β	C
value	1	48	$\mathbf{W}_{LSH}$	0	$0.4 \sim 0.5$	1

*1) **Effect of Dissimilarity Ratio Threshold** β:* As indicated in Eq. (3), β is used to control the similarity loss on dissimilar data. A large β means that dissimilar data should be critically separated as far as possible in Hamming space. Thus, a too large β may lead to excessively frequent update of the proposed OH models. In contrast, a small β indicates dissimilar data are less critically separated. Therefore, a too small β may make the hash model less discriminative. Consequently, a proper β should be set for the proposed online hashing models.

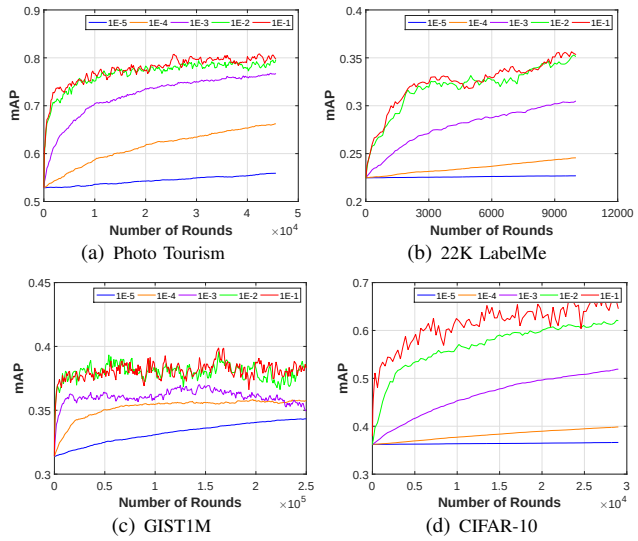
We investigate the influence of β on OH on different datasets by varying β from 0.1 to 0.9. Fig. 2 presents the experimental results. On all datasets, when β increases, the performance of OH becomes better and better at first. However, when β is larger than 0.5, further increasing β may lead to performance degradation.

In summary, a moderately large β is better on the other three datasets. Based on the experimental results, $\beta = 0.5$ is set as a default value for OH on Photo Tourism and 22K LabelMe datasets in the experiments below, and $\beta = 0.4$ is the default value for OH tested on GIST 1M and CIFAR-10 datasets.

*2) **Effect of the Number of Multiple Models** T:* Before the investigation about the effect of the number of models T on MMOH, it should be noticed that when $T = 1$, MMOH degrades to OH. And when $T > 1$, we use the multi-index technique [65] to realize fast hash search. The influence of T on MMOH is observed by varying T from 1 to 4. Fig. 3 presents the experimental results of MMOH on the four datasets. From this figure, we can find that when more models are used (i.e., larger T), the performance of MMOH on most datasets except CIFAR-10 is always better. On 22K LabelMe and GIST1M, the improvement of using more models are more clear, and it is less on Photo Tourism. On CIFAR-10, MMOH seems not sensitive to different values of T. In summary, the results in Fig. 3 suggest that MMOH performs overall better and more stably when more models are used.

*3) **Effect of Margin Parameter** C :* In Eq. (10), C is the upper bound of the τ^t, and τ^t can be viewed as the step size of the update. A large C means that $\mathbf{W}^t$ will be updated more towards reducing the loss on the current data pair. In Theorem 2, a large C is necessary to guarantee the bound on the accumulative loss for OH. In this experiment, the lower bound of C in Theorem 2 was 0.017 since the maximum value $\sqrt{R(\mathbb{h}^t, s^t)}$ was 8 and the minimum value of F^2 was 479.

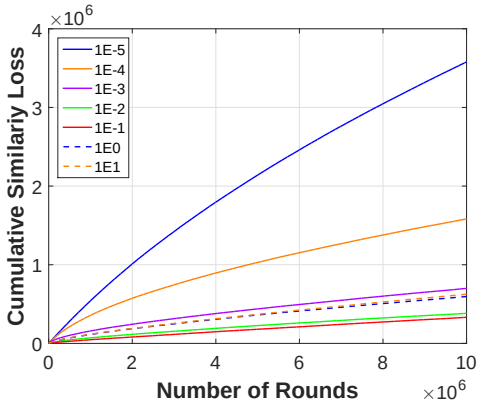
The effect of C on the mAP performance was also evaluated on all datasets by varying C from $1E{-}5$ to $1E{-}1$. From Figure 4, we find that a larger C (i.e. $C = 1E-1$) is preferred on all datasets. When C increases from $C = 1E-5$ to $C = 1E-2$, the final performance of OH improves by a large margin on all



(a) Photo Tourism (b) 22K LabelMe

(c) GIST1M (d) CIFAR-10

Fig. 4. mAP comparison results of OH with C ranging from 0.00001 to 0.1. (Best viewed in color.)



(a) Photo Tourism

Fig. 5. Cumulative similarity loss of OH with C ranging from 0.00001 to 10. (Best viewed in color.)

datasets. Further increasing the value of C can only slightly improve the final performance, especially on Photo Tourism, 22K LabelMe and Gist1M. We also find that the performance would nearly not change when setting $C > 1E-1$ as this value is larger than $\frac{\ell^t(\mathbf{W}^t)}{||\mathbf{x}^t(\mathbf{g}^t-\mathbf{h}^t)^T||_F^2}$ for most t we have investigated. Hence, $C = 1E-1$ is chosen as a default value in the other experiments for OH.

Finally, we evaluated the effect of C on the cumulative similarity loss experimentally. For this purpose, we took Photo Tourism dataset as example and ran OH over more than 10^6 loops by varying C from $1E-5$ to $1E1$, where duplication of training data was allowed. The comparison result in Figure 5 shows how the cumulative similarity loss increases when the number of iteration rounds increases. The lowest cumulative loss is achieved by setting $C = 0.1$, which is the one larger but closest to the required lower bound value of C (i.e., 0.017). When C is too small, i.e. $C < 1E-3$, the cumulative loss grows strongly. This verifies that C should be lower bounded in order to make the cumulative loss under control. If $C \geq 1E-3$, the cumulative loss tends to increase much more slowly as shown in Figure 5.

C. Comparison with Related Methods

To comprehensively demonstrate the efficiency and effectiveness of the proposed OH and MMOH, we further compared it with several related hashing algorithms in terms of mAP and training time complexity.

*1) **Compared Methods**:* There is not much work on developing online hashing methods. To make a proper comparison, the following three kinds of hashing methods were selected:

(a) KLSH [9], a variant of LSH [6] which uses RBF kernel function to randomly sample the hash projection matrix $\mathbf{W}$, is selected as the baseline comparison algorithm. This algorithm is a data-independent hashing method and thus considered as the baseline method.

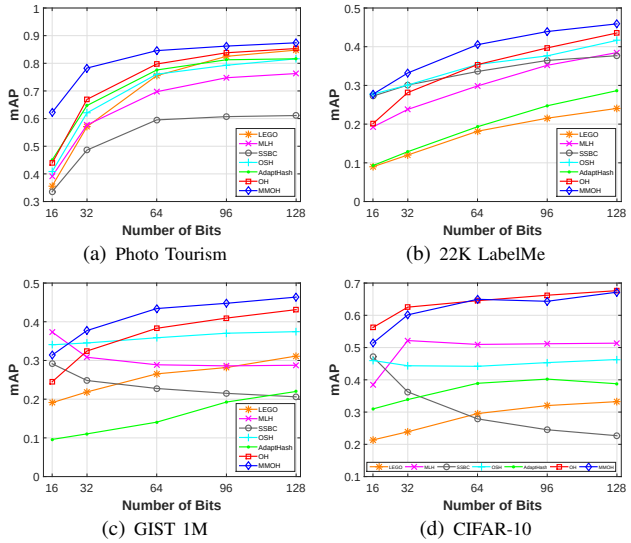
(b) Five non-batch based hashing models were selected: LEGO-LSH [41], MLH [22], SSBC [48], OSH [47] and AdaptHash [50]. LEGO-LSH [41], another variant of LSH, utilizes an online metric learning to learn a metric matrix to process online data, which does not focus on the hash function learning but on metric learning. MLH [22] and AdaptHash [50] both enable online learning by applying stochastic gradient descent (SGD) to optimize the loss function. SSBC [48] and OSH [47] are specially designed for coping with stream data by applying matrix sketching [53] on existing batch mode hashing methods. Except AdaptHash and OSH, other three methods all require that the input data should be zero-centered, which is impractical for online learning since one cannot have all data samples observed in advance.

(c) Four batch mode learning hashing models were selected. One is the unsupervised method ITQ [19]. ITQ is considered as a representative method for unsupervised hashing in batch mode. Since OH is a supervised method, we selected three supervised methods for comparison in order to see how an online approach approximate these offline approaches. The three supervised methods are supervised hashing with kernel (KSH) [21], fast supervised hash (FastHash) [26] and supervised discrete hashing (SDH) [16]. KSH is a famous supervised hashing method to preserve the pairwise similarity between data samples. FastHash and SDH are two recently developed supervised hashing method in batch mode, where we run SDH for fast search the same as in [66].

*2) **Settings**:* Since MLH, LEGO-LSH and OH are based on pairwise data input, we randomly sampled data pairs in sequence from the training set. For OSH, the chunk size of streaming data was set as 1000 and each data chunk was sequentially picked up from the training sequence as well. Additionally, the key parameters in the compared methods were set as the ones recommended in the corresponding papers. For the proposed OH, the parameters were set according to Table I and the number of models T is set to be 4 for MMOH.

*3) **Comparison with Related Online Methods**:* We compare OH with the selected related methods in two aspects: mAP comparison and training time comparison.

- mAP Comparisons



(a) Photo Tourism (b) 22K LabelMe

(c) GIST 1M (d) CIFAR-10

Fig. 6. mAP comparison results among different online hash models with respect to different code lengths. (Best viewed in color.)

Figure 6 presents the comparison results among LEGO-LSH, MLH, SSBC, OSH, OH and MMOH with the code length varying from 16 to 128.

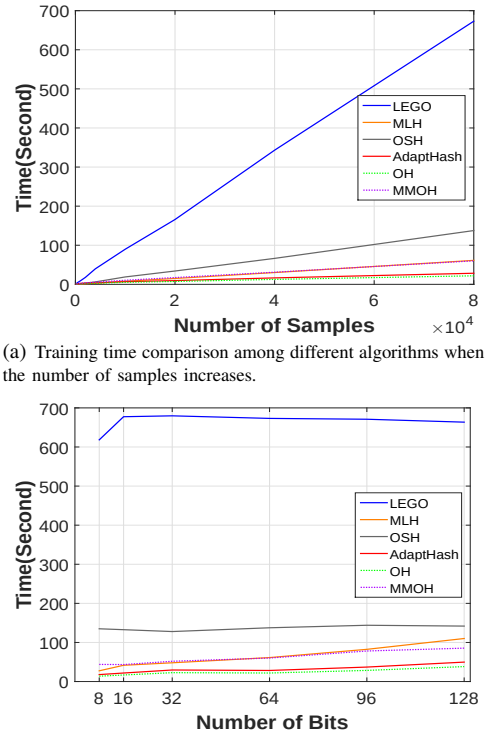
From this figure, we find that when the hash code length increases, the performance of MMOH and OH becomes better and better on all datasets. MMOH achieves the best performance and OH achieves the second on almost all datasets when the code length is larger than 32, except Photo Tourism where LEGO-LSH and OH perform very similarly. Specifically, as the bit length increases, MMOH performs significantly better than OSH and MLH on all datasets, and it performs better than LEGO-LSH on 22K LabelMe, GIST1M and CIFAR-10. In addition, as the hash bit length increases, it is more clear to see MMOH performed better than the compared methods on CIFAR-10. All these observations demonstrate the efficiency and effectiveness of OH and MMOH to the compared hashing algorithms on processing stream data in an online way.

- Training Time Comparisons

In this part, we investigate the comparison on the accumulated training time, which is another main concern for online learning. Without loss of generality, the Photo Tourism dataset was selected and 80K training points were randomly sampled to observe the performance of different algorithms. All results are averaged over 10 independent runs conducted on a server with Intel Xeon X5650 CPU, 12 GB memory and 64-bit CentOS system. For fairness, in each run, all experiments were conducted using only one thread. Specially, SSBC is excluded in this experiment since its training time is significantly longer than any other methods.

First, we investigate the training time of different algorithms when the number of training samples increases. In this experiment, the hash bit length was fixed to 64 and the comparison result is presented in Figure 7(a). From this figure, we find that:

1) As the number of training samples increases, the accumulated training time of all methods almost increases linearly. However, it is evident that LEGO-LSH increases much faster than the other compared methods. In particular, OH, AdaptHash and MMOH increase the lowest, and in particular OH takes only 0.0015 seconds (using single thread run in a single CPU) for each pair for update.
2) Compared with LEGO-LSH and OSH, the accumulated training time of other three methods are considerably much smaller. Specifically, when the number of samples is 8×10^4, the accumulated training time of LEGO-LSH is 10 times more as compared to MMOH and MLH. Besides, the time of OSH is 2 times more as compared to MMOH and MLH and is 4 times more as compared to OH and AdaptHash.
3) The training time of MMOH and MLH is very similar. The training time of OH is slightly less than AdaptHash, and it is always the least one.



(a) Training time comparison among different algorithms when the number of samples increases.

(b) Training time comparison among different algorithms with different code lengths.

Fig. 7. Training time comparison among different algorithms on Photo Tourism. (Best viewed in color.)

Second, we further investigate the training time of different methods when the hash code length increases, where the training sample size is fixed to be 80,000. The comparison result is displayed in Figure 7(b). From this figure, we find that:

1) When the code length increases, the accumulated training time of all algorithms increases slightly except LEGO-LSH. The training time of LEGO-LSH does not change obviously when the code length is larger than 24. This is because most of the training time was spent on the metric training, which is independent of the hash code length, while the generation of hash projection matrix in LSH costs very little time [41].

2) MMOH took considerably smaller accumulated training time than LEGO-LSH and OSH. This is because only a small part of hash bits are updated in MMOH, which reduces the time cost in updating.
3) Compared to MLH, MMOH took a little more time than MLH when the bit length is smaller than 64. However, when the bit length is larger than 64, the training time of MMOH becomes slightly less than that of MLH. This phenomenon can be ascribed to the fact that the time spent on selecting the hash model to preserve in MMOH does not heavily depend on the code length, and the time cost of updating hash model in MMOH grows slower than that in MLH.
4) OH took the least training time in all the comparison. The training time of AdaptHash is slightly higher. And the training time of OH and AdaptHash is only about half of the training time of MLH and MMOH.

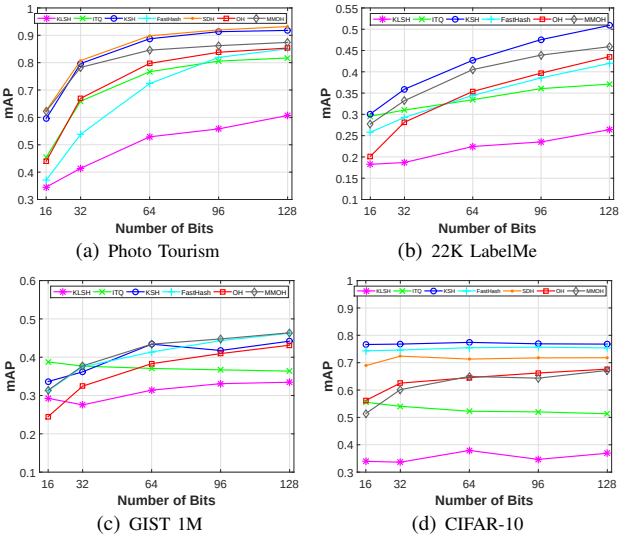
From the above investigation, we can conclude that OH and MMOH are very efficient in consuming considerably small accumulated training time and thus are more suitable for online hash function learning.

*4) **Comparison to Batch Mode Methods**:* Finally, we compare batch mode (offline) learning-based hashing models. The main objective here is not to show which is better, as online learning and offline learning have different focuses. The comparison here is to show how well an online model can now approach the largely developed batch mode methods.

Figure 8 presents the comparison results on different hash code length varying from 16 to 128 on the four datasets. SDH is only conducted on Photo Tourism and CIFAR-10 as only these two datasets provides class label information. From this figure, it is evident that OH significantly outperforms KLSH when using different numbers of hash bits on all datasets. Specifically, when the code length increases, the difference between OH and KLSH becomes much bigger, especially on Photo Tourism and 22K LabelMe.

When compared with the unsupervised methods ITQ, OH outperforms it when the code length is larger than 32 bits over all datasets. These comparison results demonstrate that OH is better than the compared data-independent methods KLSH and the data-dependent unsupervised methods ITQ overall.

For comparison with three supervised methods, the mAP of MMOH is the highest on GIST 1M dataset, slightly higher than KSH and FastHash, when the code length is larger than 16 bits. KSH and FastHash have their limitation on this dataset. KSH cannot engage all data in training as the similarity matrix required by KSH is of tremendous space cost, and the Block Graph Cut technique in FastHash may not work well on this dataset as the supervised information is generated by the Euclidean distance between data samples. On the other three datasets, MMOH is not the best, but performs better than two unsupervised methods KLSH and ITQ. Indeed, from the performance aspect, when all labeled data are available at one time using OH and MMOH is not the best choice as compared to batch mode supervised hashing methods. However, OH and MMOH solve a different problem as compared to the supervised batch ones. OH and MMOH concern how to train supervised hashing model on stream data in an online learning framework, while the batch ones are not. A bound on the cumulative loss function is necessary for an online learning model while it is not necessary for batch mode methods. Hence, OH and MMOH have their unique merits.



(a) Photo Tourism (b) 22K LabelMe (c) GIST 1M (d) CIFAR-10

Fig. 8. mAP comparison against KLSH and learning-based batch mode methods with different code lengths on all datasets. (Best viewed in color.)

VIII. Conclusion & Discussion

In this paper, we have proposed an one-pass online learning algorithm for hash function learning called Online Hashing. OH updates its hash model in every step based on a pair of input data samples. We first re-express the hash function as a form of structured prediction and then propose a prediction loss function according to it. By penalizing the loss function using the previously learned model, we update the hash model constrained by an inferred optimal hash codes which achieve zero prediction loss. The proposed online hash function model ensures that the updated hash model is suitable for the current pair of data and has theoretical upper bounds on the similarity loss and the prediction loss. Finally, a multi-model online hashing (MMOH) is developed for a more robust online hashing. Experimental results on different datasets demonstrate that our approach gains satisfactory results, both efficient on training time and effective on the mAP results. As part of future work, it can be expected to consider updating the model on multiple data pairs at one time. However, the theoretical bound for online learning needs further investigation in this case.

Acknowledgements

This work was completed when the first student was an undergraduate student at Sun Yat-sen University.

References

[1] A. R. Webb and K. D. Copsey, *Statistical pattern recognition*. Wiley. com, 2003.

[2] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. J. Russell, "Distance metric learning, with application to clustering with side-information," in *Neural Information Processing Systems*, 2002, pp. 521–528.

[3] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Neural Information Processing Systems*, 2005, pp. 1473–1480.

[4] W.-S. Zheng, S. Gong, and T. Xiang, "Re-identification by relative distance comparison," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 3, pp. 653–668, 2013.

[5] M. Norouzi, A. Punjani, and D. Fleet, "Fast exact search in hamming space with multi-index hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 6, pp. 1107–1119, 2014.

[6] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *ACM Symposium on Theory of Computing*, 2002, pp. 380–388.

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

[8] O. Chum, J. Philbin, and A. Zisserman, "Near duplicate image detection: min-hash and tf-idf weighting," in *British Machine Vision Conference*, 2008, pp. 1–10.

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

[10] Z. Tang, X. Zhang, and S. Zhang, "Robust perceptual image hashing based on ring partition and nmf," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 3, pp. 711–724, 2014.

[11] L. Zhang, Y. Zhang, X. Gu, J. Tang, and Q. Tian, "Scalable similarity search with topology preserving hashing," *IEEE Transactions on Image Processing*, vol. 23, no. 7, pp. 3025–3039, 2014.

[12] A. Shrivastava and P. Li, "Densifying one permutation hashing via rotation for fast near neighbor search," in *International Conference on Machine Learning*, 2014, pp. 557–565.

[13] Z. Yu, F. Wu, Y. Zhang, S. Tang, J. Shao, and Y. Zhuang, "Hashing with list-wise learning to rank," in *ACM Special Interest Group on Information Retrieval*, 2014, pp. 999–1002.

[14] X. Liu, J. He, and B. Lang, "Multiple feature kernel hashing for large-scale visual search," *Pattern Recognition*, vol. 47, no. 2, pp. 748–757, 2014.

[15] B. Wu, Q. Yang, W. Zheng, Y. Wang, and J. Wang, "Quantized correlation hashing for fast cross-modal search," in *International Joint Conference on Artificial Intelligence*, 2015, pp. 3946–3952.

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

[17] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Neural Information Processing Systems*, 2008, pp. 1753 –1760.

[18] W. Liu, J. Wang, S. Kumar, and S. Chang, "Hashing with graphs," in *International Conference on Machine Learning*, 2011, pp. 1–8.

[19] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2011, pp. 817–824.

[20] J. Heo, Y. Lee, J. He, S. Chang, and S. Yoon, "Spherical hashing," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2957–2964.

[21] W. Liu, J. Wang, R. Ji, Y. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 2074–2081.

[22] M. Norouzi and D. Fleet, "Minimal loss hashing for compact binary codes," in *International Conference on Machine Learning*, 2011, pp. 353–360.

[23] Y. Mu, J. Shen, and S. Yan, "Weakly-supervised hashing in kernel space," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3344 –3351.

[24] P. Zhang, W. Zhang, W.-J. Li, and M. Guo, "Supervised hashing with latent factor models," in *ACM Special Interest Group on Information Retrieval*, 2014, pp. 173–182.

[25] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *The Association for the Advancement of Artificial Intelligence*, 2014, pp. 2156–2162.

[26] G. Lin, C. Shen, Q. Shi, A. v. d. Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1971–1978.

[27] J. Wang, O. Kumar, and S. Chang, "Semi-supervised hashing for scalable image retrieval," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2010, pp. 3424 –3431.

[28] J. Wang, S. Kumar, and S. Chang, "Sequential projection learning for hashing with compact codes," in *International Conference on Machine Learning*, 2010, pp. 1127–1134.

[29] 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.

[30] N. Quadrianto and C. H. Lampert, "Learning multi-view neighborhood preserving projections," in *International Conference on Machine Learning*, 2011, pp. 425–432.

[31] M. Rastegari, J. Choi, S. Fakhraei, D. Hal, and L. Davis, "Predictable dual-view hashing," in *International Conference on Machine Learning*, 2013, pp. 1328–1336.

[32] D. Zhang and W.-J. Li, "Large-scale supervised multimodal hashing with semantic correlation maximization," in *The Association for the Advancement of Artificial Intelligence*, 2014, pp. 2177–2183.

[33] J. Masci, M. Bronstein, A. Bronstein, and J. Schmidhuber, "Multimodal similarity-preserving hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 4, pp. 824–830, 2014.

[34] Y. Wei, Y. Song, Y. Zhen, B. Liu, and Q. Yang, "Scalable heterogeneous translated hashing," in *ACM Special Interest Group on Knowledge Discovery and Data Mining*, 2014, pp. 791–800.

[35] D. Wang, X. Gao, X. Wang, and L. He, "Semantic topic multimodal hashing for cross-media retrieval," in *International Joint Conference on Artificial Intelligence*, 2015, pp. 3890–3896.

[36] D. Wang, X. Gao, X. Wang, L. He, and B. Yuan, "Multimodal discriminative binary embedding for large-scale cross-modal retrieval," *IEEE Transactions Image Processing*, vol. 25, no. 10, pp. 4540–4554, 2016.

[37] Y. Zhen and D.-Y. Yeung, "Active hashing and its application to image and text retrieval," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 255–274, 2013.

[38] Q. Wang, L. Si, Z. Zhang, and N. Zhang, "Active hashing with joint data example and tag selection," in *ACM Special Interest Group on Information Retrieval*. ACM, 2014, pp. 405–414.

[39] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551–585, 2006.

[40] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, "Large scale online learning of image similarity through ranking," *Journal of Machine Learning Research*, vol. 11, pp. 1109–1135, 2010.

[41] P. Jain, B. Kulis, I. S. Dhillon, and K. Grauman, "Online metric learning and fast similarity search," in *Neural Information Processing Systems*, 2008, pp. 761–768.

[42] Y. Li and P. M. Long, "The relaxed online maximum margin algorithm," in *Neural Information Processing Systems*, 1999, pp. 498–504.

[43] M. K. Warmuth and D. Kuzmin, "Randomized online pca algorithms with regret bounds that are logarithmic in the dimension," *Journal of Machine Learning Research*, 2008.

[44] J. Silva and L. Carin, "Active learning for online bayesian matrix factorization," in *ACM Conference on Knowledge Discovery and Data Mining*, 2012, pp. 325–333.

[45] A. Bordes, S. Ertekin, J. Weston, and L. Bottou, "Fast kernel classifiers with online and active learning," *Journal of Machine Learning Research*, vol. 6, pp. 1579–1619, 2005.

[46] W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. Tseng, "Unbiased online active learning in data streams," in *ACM Conference on Knowledge Discovery and Data Mining*, 2011, pp. 195–203.

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

[48] M. Ghashami and A. Abdullah, "Binary coding in stream," *CoRR*, vol. abs/1503.06271, 2015.

[49] F. Çakir and S. Sclaroff, "Online supervised hashing," in *IEEE International Conference on Image Processing*, 2015, pp. 2606–2610.

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

[51] ——, "Supervised hashing with error correcting codes," in *ACM International Conference on Multimedia*, 2014, pp. 785–788.

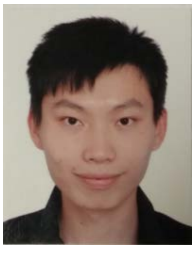
[52] L.-K. Huang, Q. Yang, and W.-S. Zheng, "Online hashing," in *International Joint Conferences on Artificial Intelligence*, 2013.

[53] E. Liberty, "Simple and deterministic matrix sketching," in *ACM Conference on Knowledge Discovery and Data Mining*, 2013, pp. 581–588.

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

[55] T. Finley and T. Joachims, "Training structural svms when exact inference is intractable," in *International Conference on Machine Learning*, 2008, pp. 304–311.

[56] T. H. Ioannis Tsochantaridis, Thorsten Joachims and Y. Altun, "Large margin methods for structured and interdependent output variables," *Journal of Machine Learning Research*, vol. 6, pp. 1453–1484, 2005.
[57] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo tourism: Exploring photo collections in 3d," in *ACM Transactions on Graphics*, vol. 25, no. 3, 2006, pp. 835–846.
[58] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2008, pp. 1–8.
[59] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011.
[60] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.
[61] Z. Jin, C. Li, Y. Lin, and D. Cai, "Density sensitive hashing," *IEEE Transactions on Cybernetics*, vol. 44, no. 8, pp. 1362–1371, 2014.
[62] W. F. A. Torralba, R. Fergus and C. MIT, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, 2008.
[63] A. Turpin and F. Scholer, "User performance versus precision measures for simple search tasks." in *ACM Special Interest Group on Information Retrieval*, 2006, pp. 11–18.
[64] 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.
[65] M. Norouzi, A. Punjani, and D. J. Fleet, "Fast search in hamming space with multi-index hashing," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 3108–3115.
[66] L. Huang and S. J. Pan, "Class-wise supervised hashing with label embedding and active bits," in *International Joint Conferences on Artificial Intelligence*, 2016.



Long-Kai Huang is pursuing the Ph.D degree in School of Computer Science and Engineering, Nanyang Technological University, Singapore. He received his B.E. degree from Information Science and Technology, Sun Yat-Sen University, Guangzhou, China in 2013. His currect researh interests are in machine learning and computer vision, and specially focus on fast large-scale image retrieval, non-convex optimization and its applications.

Qiang Yang (S14) received his M. S. degree from the School of Infomration Science and Technology, Sun Yat-sen University, China, in 2014. Currently, he is pursuing his Ph. D. degree at the School of Data and Computer Science, Sun Yat-sen University, China. His current research interests include data mining algorithms, machine learning algorithms, evolutionary computation algorithms and their applications on real-world problems.

Wei-Shi Zheng is now a Professor at Sun Yat-sen University. He has now published more than 90 papers, including more than 60 publications in main journals (TPAMI,TIP,PR) and top conferences (ICCV, CVPR,IJCAI). His research interests include person/object association and activity understanding in visual surveillance. He has joined Microsoft Research Asia Young Faculty Visiting Programme. He is a recipient of Excellent Young Scientists Fund of the NSFC, and a recipient of Royal Society-Newton Advanced Fellowship. Homepage: http://isee.sysu.edu.cn/%7ezhwshi/