Online Collective Matrix Factorization Hashing for Large-Scale Cross-Media Retrieval

Di Wang
Xidian University
wangdi@xidian.edu.cn

Quan Wang
Xidian University
qwang@xidian.edu.cn

Yaqiang An
Xidian University
gnaiqayna@gmail.com

Xinbo Gao*
Xidian University
xbgao@mail.xidian.edu.cn

Yumin Tian
Xidian University
ymtian@mail.xidian.edu.cn

ABSTRACT

Cross-modal hashing has been widely investigated recently for its efficiency in large-scale cross-media retrieval. However, most existing cross-modal hashing methods learn hash functions in a batch-based learning mode. Such mode is not suitable for large-scale data sets due to the large memory consumption and loses its efficiency when training streaming data. Online cross-modal hashing can deal with the above problems by learning hash model in an online learning process. However, existing online cross-modal hashing methods cannot update hash codes of old data by the newly learned model. In this paper, we propose Online Collective Matrix Factorization Hashing (OCMFH) based on collective matrix factorization hashing (CMFH), which can adaptively update hash codes of old data according to dynamic changes of hash model without accessing to old data. Specifically, it learns discriminative hash codes for streaming data by collective matrix factorization in an online optimization scheme. Unlike conventional CMFH which needs to load the entire data points into memory, the proposed OCMFH retrains hash functions only by newly arriving data points. Meanwhile, it generates hash codes of new data and updates hash codes of old data by the latest updated hash model. In such way, hash codes of new data and old data are well-matched. Furthermore, a zero mean strategy is developed to solve the mean-varying problem in the online hash learning process. Extensive experiments on three benchmark data sets demonstrate the effectiveness and efficiency of OCMFH on online cross-media retrieval.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; *Retrieval models and ranking*; Top-k retrieval in databases.

KEYWORDS

Cross-Modal Hashing; Large-Scale Retrieval; Online Learning; Matrix Factorization.

ACM Reference Format:
Di Wang, Quan Wang, Yaqiang An, Xinbo Gao, and Yumin Tian. 2020. Online Collective Matrix Factorization Hashing for Large-Scale Cross-Media Retrieval. In *43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20), July 25–30, 2020, Virtual Event, China.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3397271.3401132

*Corresponding author

SIGIR '20, July 25–30, 2020, Virtual Event, China

ACM ISBN 978-1-4503-8016-4/20/07…$15.00
https://doi.org/10.1145/3397271.3401132

1 INTRODUCTION

With the explosive growth of multimedia data from various social media websites, cross-modal hashing (CMH) has gained much attentions due to its exclusive capability of facilitating large-scale cross-media retrieval [14], [19], [6]. The goal of CMH is to project high-dimensional data with different modalities (e.g. images, texts, videos) into compact binary codes that preserve the intrinsic structures of the original data through a series of hashing functions. By doing so, Hamming distances between binary codes can be applied as the metric to measure cross-modality similarities. With fast bit-wise operation on binary hash codes, CMH gains significantly computational savings both in speed and storage for large-scale data sets.

In recent years, many CMH methods have been proposed and have achieved promising performance [20], [30], [12], [29], [13]. Although substantial progress has been made in CMH, most existing methods are batch-based methods. Such methods learn hash functions in offline processes and need to collect all training data points in advance. Therefore, they suffer from two critical problems in the real-word applications. First, they cannot effectively deal with streaming data. In real world online web cross-media retrieval, multimedia data points usually continuously arrive in a streaming fashion. For example, social media websites like Facebook, Instagram and Twitter generate huge amounts of data every day. After new data is arriving, batch-based methods should retrain hash models on all the accumulated data and recompute hash codes of the entire data points, which are obviously inefficient. Second, they have unaffordable computation and memory cost when dealing with truly large-scale data sets. As the growing amount of data, it is unrealistic to load all the data into memory to learn hash functions, due to the large memory overhead. Even if the memory is large

enough to load all the data for these methods, the high computation burden and excessively long computational time are often intolerable.

To meet the aforementioned challenges, online CMH methods are proposed to learn hashing models in online learning modes by processing one chunk of data points at a time. Such methods incrementally update hash functions from sequentially arriving data to adapt to the variations of data stream, and simultaneously encode new data into compact binary codes. Therefore, they can effectively cope with streaming data. And meanwhile, since hash functions are updated only by current data, online CMH methods greatly reduce the computation cost and memory requirement than batch-based methods. Due to these merits, online CMH arouses great attention very recently and several methods have been proposed. According to whether supervised information is used, online CMH methods can be grouped into two categories: supervised methods and unsupervised methods.

Supervised online CMH methods take advantage of supervised information such as labels to learn semantic preserving hash codes for streaming data. Online latent semantic hashing (OLSH) [28] maps discrete labels into a continuous latent semantic space to obtain hash codes, and utilizes an online optimization scheme to learn hash model. Flexible online multi-modal hashing (FOMH) [15] develops an asymmetric supervised online hashing model to project multimodal data into hash codes. And it exploits a self-weighted modality fusion strategy to adaptively learn weight for each modality for online streaming multimodal data. Despite supervised methods can achieve good performance, supervised information of the entire data set is difficult to obtain since labeling samples requires much human effort.

Unsupervised online CMH methods learn hash codes from data distribution and are more applicable than supervised ones in practical applications. To the best of our knowledge, online cross-modal hashing (OCMH) [27] and dynamic multi-view hashing (DMVH) [26] are only two unsupervised online cross-modal hashing methods at present. OCMH represents hash codes as permanent shared latent codes and dynamic transfer matrix. Permanent hash codes are generated by current arriving data to preserve information of current data. Dynamic transfer matrix is incrementally updated upon the arrival of streaming data. Whereas, shared latent codes of OCMH are learned by current data. They are immutable as hash model changes, which lead to mismatching between hash codes of old and new data. Despite OCMH attempts to modify the mismatching problem by dynamic transfer matrix, the mismatching still exists. DMVH constructs dictionary to represent multimodal data and supports online hash codes generation. It dynamically augments hash code length with the changes of streaming data. However, hash codes generated by DMVH usually have high redundancy and long bit length especially when newly arriving data is always different to old data. This is because DMVH augments hash codes with ever-changing of data. In addition, DMVH also cannot efficiently update hash codes of old data by the latest updated model. In light of the above analysis, we can find that none of the existing unsupervised online CMH methods can effectively update hash codes of old data with the changes of hash model as far as we know.

In this paper, we propose a novel unsupervised online cross-modal hashing method named online collective matrix factorization hashing (OCMFH) based on collective matrix factorization hashing (CMFH), to adaptively update hash codes of old data with changes of hash model without referring to original old data points. As illustrated in Figure 1, the proposed OCMFH incrementally updates hash functions by the current arriving data and generates hash codes for them. After hash model is updated, OCMFH tunes hash codes of old data to adapt to hash model changes. Therefore, hash codes of old and new data are well-matched. Contributions of the proposed OCMFH are listed as following:

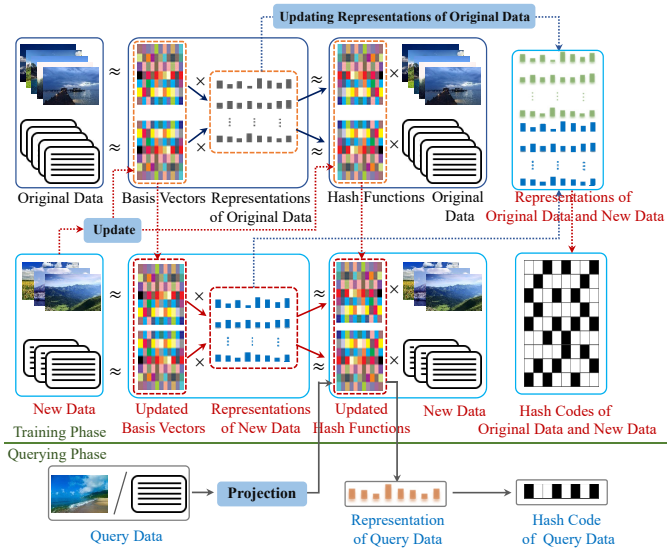

Figure 1: Framework of online collective matrix factorization hashing.

- The proposed OCMFH extends CMFH to an online learning mode by developing an efficient online optimization method. It incrementally updates hash functions to adapt to the variations of streaming multi-modal data and simultaneously generates hash codes for the current arriving data. Unlike conventional CMFH which needs to load the entire data points into memory, OCMFH processes only one chunk of data points at a time and thus greatly reduces the computational overhead and memory requirement.
- The proposed OCMFH can dynamically update hash codes of old data with the changes of hash model without accessing to original old data. Consequently, hash codes of old data and new data will be well-matched and the retrieval performance will be enhanced.
- A zero-mean strategy is developed to overcome the mean-varying problem in online hash learning process.

The rest of this paper is organized as follows. Section 2 briefly reviews several existing online hashing algorithms and the closely related collective matrix factorization hashing method. Section 3 introduces the proposed online collective matrix factorization hashing method and its theoretical analysis. Section 4 provides extensive experimental results on three benchmark data sets. Finally, the conclusions are reached in Section 5.

2 RELATED WORK

2.1 Online Hashing

Online hashing incrementally learns hash model by processing data in a sequential order with one chunk of data points at a time [17], [18]. Compared with traditional batch-based hashing methods, they have much lower computation and memory costs when processing large-scale data sets and can efficiently deal with streaming data, thus attract much attention very recently.

Early efforts at online hashing are single-modal methods. Such methods aim at learning hash codes for unimodal data. According to the ways of learning hash models for streaming data, single-modal online hashing methods can be roughly categorized as stochastic gradient decent (SGD)-based methods and sketching-based methods [25]. SGD-based methods first construct similarity-preserving objective functions and then utilize SGD to online optimize the objective functions. Representative methods are online kernel-based hashing (OKH) [7], online supervised hashing (OSupH) [1], mutual information hashing (MIHash) [2], and Hadamard codebook based online hashing (HCOH) [10]. Sketching-based methods first sketch streaming data in small-size data sketches which maintain main information of original data sets. Then they learn hash functions in online modes by data sketches. Online sketching hashing (OSH) [9], online supervised sketching hashing (OSSH) [25], and faster online sketching hashing (FROSH) [3] are representative sketching-based single-modal online hashing methods.

Online cross-modal hashing is designed for data with multiple modalities. Only a few online cross-modal hashing methods are proposed at present. According to whether supervised information is utilized, they can be divided into supervised ones and unsupervised ones. Supervised ones take advantage of similarity information provided by semantic labels to learn discriminative hash codes. Such as online latent semantic hashing (OLSH) [28] and flexible online multi-modal hashing (FOMH) [15]. Online multi-modal hashing (OMMH) [16] is another kind of supervised cross-modal hashing method. It trains hash model in a batch-based mode and supports online hashing on query stage. It adaptively learns query hash code according to the dynamic query content. Generally, supervised methods can achieve promising performance. However, supervised information of the entire data set is difficult to obtain since labeling samples requires much human effort. Unsupervised online cross-modal hashing incrementally learns hash functions from data distribution. They are more applicable than supervised ones. To the best of our knowledge, there are only two unsupervised online cross-modal hashing methods at present, i.e. online cross-modal hashing (OCMH) [27] and dynamic multi-view hashing (DMVH) [26]. Whereas, OCMH and DMVH cannot effectively update hash codes of old data without retraining the hash model or accessing to old data points. Focusing on this problem, we propose a new online cross-modal hashing method named online collective matrix factorization hashing in this paper.

2.2 Collective Matrix Factorization Hashing

The proposed online collective matrix factorization hashing method is an online version of collective matrix factorization hashing (CMFH) [5]. In this section, we briefly review CMFH.

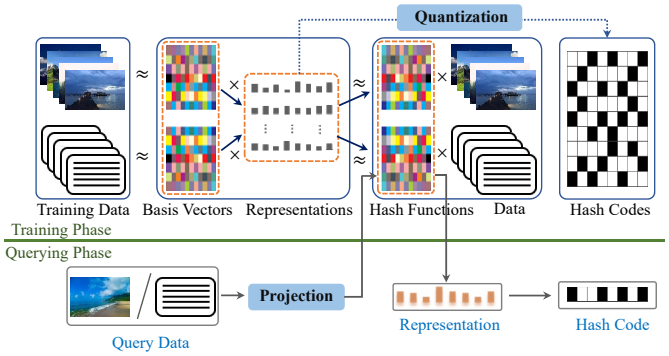

Figure 2: Framework of collective matrix factorization hashing.

Algorithm 1 Collective matrix factorization hashing

Training stage

Input: Data matrices $\mathbf{X}_i$, $i = 1, 2, \cdots s$, parameters λ_i, μ, and γ, hash code length k.

Output: Hash codes $\mathbf{B}$, projection matrices $\mathbf{P}_i$, $i = 1, 2, \cdots s$.

Procedure:

1. Centering $\mathbf{X}_i$ by means, $i = 1, 2, \cdots s$.

2. Initialize $\mathbf{V}$ by random matrix.

repeat

3.1 Update $\mathbf{U}_i$ by $\mathbf{U}_i = \mathbf{X}_i\mathbf{V}^T\left(\mathbf{V}\mathbf{V}^T + \gamma/\lambda_i\mathbf{I}\right)^{-1}$, $i = 1, 2, \cdots s$.

3.2 Update $\mathbf{P}_i$ by $\mathbf{P}_i = \mathbf{V}\mathbf{X}_i^T\left(\mathbf{X}_i\mathbf{X}_i^T + \gamma/\mu\mathbf{I}\right)^{-1}$, $i = 1, 2, \cdots s$.

3.3 Update $\mathbf{V}$ by $\mathbf{V} = \left(\sum_i \lambda_i\mathbf{U}_i^T\mathbf{U}_i + (2\mu + \gamma\mathbf{I})\right)^{-1}\left(\sum_i \left(\lambda_i\mathbf{U}_i^T + \mu\mathbf{P}_i\right)\mathbf{X}_i\right)$.

until convergence or reaching the maximum iteration.

4. Calculate hash codes $\mathbf{B}$ by $\mathbf{B} = \mathrm{sgn}(\mathbf{V})$.

Querying stage

Input: Feature vector $\mathbf{x}_i$, projection matrices $\mathbf{P}_i$, $i = 1, 2, \cdots s$.

Output: Hash code $\mathbf{b}$.

Procedure:

1. Centering $\mathbf{x}_i$ by means.

2. Calculate hash code $\mathbf{b}$ by $\mathbf{b} = \mathrm{sgn}(\mathbf{P}_i\mathbf{x}_i)$.

Figure 2 shows the framework of CMFH. It transforms multi-modal data into low-dimensional latent semantic spaces by collective matrix factorization [22], in which a data pair shares a unified representation. For out-of-sample extension, it learns view-specific hash functions for different modalities. The objective function of CMFH is

$$\begin{aligned}&\min_{\mathbf{U}_i,\mathbf{P}_i,\mathbf{V}} \sum_i \lambda_i \left\|\mathbf{X}_i - \mathbf{U}_i\mathbf{V}\right\|_F^2 + \mu\left\|\mathbf{V} - \mathbf{P}_i\mathbf{X}_i\right\|_F^2 \\ &+\gamma\sum_i R(\mathbf{U}_i, \mathbf{P}_i) + R(\mathbf{V}) \\ &s.t. \sum_i \lambda_i = 1,\end{aligned} \tag{1}$$

where $\mathbf{X}_i \in \mathbb{R}^{d_i \times n}$ is the zero-mean feature vector of the i-th modality, d_i and n are the dimension and number of feature vectors, respectively. $\mathbf{U}_i \in \mathbb{R}^{d_i \times k}$ and $\mathbf{V} \in \mathbb{R}^{k \times n}$ are the basis vectors and unified representations, k is the hash code length. $\mathbf{P}_i \in \mathbb{R}^{k \times d_i}$ is the projection matrix of hash function, λ_i, μ, and γ are nonnegative parameters, and $R(\cdot) = \|\cdot\|_F^2$ is the regularization term to avoid overfitting. The objective function (1) can be solved by updating each matrix variables iteratively. The learning procedure of CMFH is summarized in Algorithm 1.

CMFH is a batch-based method. It needs to load the entire data points into memory to learn hash model. Therefore, it suffers from high memory and computation burden for large-scale cross-media retrieval. In addition, it cannot incrementally learn hash functions for streaming data. To solve these problems, we propose online collective matrix factorization hashing method in this paper.

3 ONLINE COLLECTIVE MATRIX FACTORIZATION HASHING

In this section, we introduce the details of the proposed online collective matrix factorization hashing method. To simplify the presentation, we first focus on OCMFH for bi-modal data consisting of images and texts, and then extend it to the multimodal case.

3.1 Notations and Problem Description

Suppose the training data set consists of streaming image-text data pairs. At each round t, a new chunk of image-text pairs $\mathbf{X}^{(t)} = \{\mathbf{X}_1^{(t)}, \mathbf{X}_2^{(t)}\}$ is added to the training set, where $\mathbf{X}_1^{(t)} \in \mathbb{R}^{d_1 \times n_t}$ and $\mathbf{X}_2^{(t)} \in \mathbb{R}^{d_2 \times n_t}$ are image and text features, respectively, n_t is the size of new data chunk, d_1 and d_2 are the dimensionalities of image and text feature. Before round t, there has accumulated an old data chunk $\tilde{\mathbf{X}}^{(t-1)} = \{\tilde{\mathbf{X}}_1^{(t-1)}, \tilde{\mathbf{X}}_2^{(t-1)}\}$ consisting of $\tilde{n}_{t-1}$ image-text pairs. After arrival of the new chunk, the total training data chunk is denoted as $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2\}$ consisting of $n = n_t + \tilde{n}_{t-1}$ data points, where $\mathbf{X}_1 = [\tilde{\mathbf{X}}_1^{(t-1)}, \mathbf{X}_1^{(t)}] \in \mathbb{R}^{d_1 \times n}$ and $\mathbf{X}_2 = [\tilde{\mathbf{X}}_2^{(t-1)}, \mathbf{X}_2^{(t)}] \in \mathbb{R}^{d_2 \times n}$.

At each round t, the proposed OCMFH aims to learn hash functions and unified hash codes for the total training data chunk. Let $\mathbf{B} = [\tilde{\mathbf{B}}^{(t-1)}, \mathbf{B}^{(t)}] \in \mathbb{R}^{k \times n}$ denote hash codes for data chunk $\mathbf{X}$, where $\mathbf{B}^{(t)} \in \mathbb{R}^{k \times n_t}$ and $\tilde{\mathbf{B}}^{(t-1)} \in \mathbb{R}^{k \times \tilde{n}_{t-1}}$ are hash codes for new data chunk $\mathbf{X}^{(t)}$ and old data chunk $\tilde{\mathbf{X}}^{(t-1)}$, respectively, and k is the hash code length. Hash functions for image and text are defined as

$$h_m(\mathbf{x}_m) = \mathrm{sgn}(\mathbf{P}_m \mathbf{x}_m - \mathbf{b}_m), \tag{2}$$

where $\mathbf{x}_m \in \mathbb{R}^{d_m}$ is a feature vector, $\mathbf{P}_m \in \mathbb{R}^{k \times d_m}$ is a projection matrix, $\mathbf{b}_m \in \mathbb{R}^k$ is a scalar threshold vector, and $m = \{1, 2\}$. Generally, $\mathbf{b}_m$ is set to the mean of the projected training data to get balanced hash codes (each bit has equal number of 1 and -1), i.e. $\mathbf{b}_m = \frac{1}{n} \sum_i \mathbf{P}_m \mathbf{x}_{m(i)}$, where $\mathbf{x}_{m(i)}$ is the i-th feature vector of image or text training data $\mathbf{X}_m$. Suppose the training data is zero-centered, then hash functions can be rewritten as:

$$h_m(\mathbf{x}_m) = \mathrm{sgn}(\mathbf{P}_m \mathbf{x}_m). \tag{3}$$

In OCMFH, we subtract the mean of data for simplicity.

The key point of OCMFH is to incrementally learn hash functions and hash codes by the current arriving data. Suppose we have learned hash functions $h_m^{(t-1)}(\mathbf{x}_m)$ and hash codes $\tilde{\mathbf{B}}^{(t-1)}$ at previous round $t-1$. Then, after a new data chunk $\mathbf{X}^{(t)}$ is arriving at round t, the goal of OCMFH is to: 1) efficiently update hash functions to fit the newly arriving data chunk and meanwhile match with the old data chunk; 2) generate hash codes $\mathbf{B}^{(t)}$ for $\mathbf{X}^{(t)}$; 3) update hash codes of old data chunk $\tilde{\mathbf{B}}^{(t-1)}$ to fit the new hash model without accessing to original old data chunk $\tilde{\mathbf{X}}^{(t-1)}$.

3.2 Zero-Mean Normalization

In OCMFH, we subtract the mean of training data for the convenience of learning hash functions. In an online setting, each data chunk is arriving at a streaming manner. The exact mean of training data cannot be calculated before hash learning process. And as data is continuously changing, the mean of data also changes. In this paper, we adopt a zero-mean normalization strategy to subtract mean of data for OCMFH.

At round $t-1$, suppose we have a data chunk $\tilde{\mathbf{X}}_m^{(t-1)}$ with mean as $\tilde{\mathbf{u}}_m^{(t-1)}$, where $m = \{1, 2\}$. Then the zero-mean data at this time is $\overline{\tilde{\mathbf{X}}}_m^{(t-1)} = \tilde{\mathbf{X}}_m^{(t-1)} - \tilde{\mathbf{u}}_m^{(t-1)}\mathbf{1}$, where $\mathbf{1}$ is a constant vector with all ones. At round t, a new data chunk $\mathbf{X}_m^{(t)}$ is arriving with mean as $\mathbf{u}_m^t$. Then the mean of the current entire data chunk will change to

$$\mathbf{u}_m = \frac{\tilde{n}_{t-1}\tilde{\mathbf{u}}_m^{(t-1)} + n_t \mathbf{u}_m^t}{\tilde{n}_{t-1} + n_t}. \tag{4}$$

So, the new zero-mean data chunk will be $\overline{\mathbf{X}}_m^{(t)} = \mathbf{X}_m^{(t)} - \mathbf{u}_m\mathbf{1}$.

In the following, we utilize zero-mean data chunk $\overline{\mathbf{X}}_m^{(t)}$ to learn hash model. For convenience, we still use $\mathbf{X}_m^{(t)}$ to represent $\overline{\mathbf{X}}_m^{(t)}$.

3.3 Model Formulation

The proposed OCMFH is an online version of CMFH. It incrementally learns hash functions and hash codes in a streaming fashion with low computation and storage complexity. Its main objective function is the same with that of CMFH. According to Section 2.2, its main objective function is defined as

$$\min_{\mathbf{U}_1, \mathbf{U}_2, \mathbf{P}_1, \mathbf{P}_2, \mathbf{V}} \mathcal{F}(\mathbf{U}_1, \mathbf{U}_2, \mathbf{P}_1, \mathbf{P}_2, \mathbf{V}), \tag{5}$$

where

$$\begin{aligned} \mathcal{F} =& \lambda \|\mathbf{X}_1 - \mathbf{U}_1\mathbf{V}\|_F^2 + (1-\lambda)\|\mathbf{X}_2 - \mathbf{U}_2\mathbf{V}\|_F^2 \\ &+ \mu(\|\mathbf{V} - \mathbf{P}_1\mathbf{X}_1\| + \|\mathbf{V} - \mathbf{P}_2\mathbf{X}_2\|) \\ &+ \gamma R(\mathbf{U}_1, \mathbf{U}_2, \mathbf{P}_1, \mathbf{P}_2, \mathbf{V}) \end{aligned}, \tag{6}$$

where $\mathbf{U}_1 \in \mathbb{R}^{d_1 \times k}$ and $\mathbf{U}_2 \in \mathbb{R}^{d_2 \times k}$ are basis vectors for images and texts, respectively. $\mathbf{V} \in R^{k \times n}$ is the unified latent representations of image-text data pairs. $\mathbf{P}_1 \in \mathbb{R}^{k \times d_1}$ and $\mathbf{P}_2 \in \mathbb{R}^{k \times d_2}$ are projection matrices of image hash function and text hash function. $\lambda \in (0, 1)$ is a parameter that controls weights of image and text, μ is a nonnegative parameter controlling the contribution of hash function learning term $\|\mathbf{V} - \mathbf{P}_1\mathbf{X}_1\| + \|\mathbf{V} - \mathbf{P}_2\mathbf{X}_2\|$. And $R(\cdot) = \|\cdot\|_F^2$ is a regularization term to avoid overfitting.

After obtaining unified representations $\mathbf{V}$, hash codes $\mathbf{B}$ can be obtained by quantifying $\mathbf{V}$ as

$$\mathbf{B} = \mathrm{sgn}(\mathbf{V}). \tag{7}$$

Therefore, we focus on learning unified representations $\mathbf{V}$ in training stage.

Figure 1 depicts the learning framework of OCMFH. When a new data chunk arrives, it first updates hash model to adapt to data variations. Then it generates hash codes of new data. Finally, it updates hash codes of old data according to the updated hash model. In the following, we first introduce how to update hash model and generate hash codes of new data in Section 3.4, then detail how to update hash codes of old data in Section 3.5.

3.4 Online Optimization

The key point of OCMFH is how to incrementally update hash model with sequential arriving data chunk. In the online learning process, at round t ($t \geq 2$), the total training data chunk $\mathbf{X} = \{\mathbf{X}_1, \mathbf{X}_2\}$ is consisting of new data chunk $\mathbf{X}^{(t)} = \{\mathbf{X}_1^{(t)}, \mathbf{X}_2^{(t)}\}$ and old data chunk $\tilde{\mathbf{X}}^{(t-1)} = \{\tilde{\mathbf{X}}_1^{(t-1)}, \tilde{\mathbf{X}}_2^{(t-1)}\}$. Then, the online formulation of objective function $\mathcal{F}$ at round t can be represented as

$$\begin{aligned}\mathcal{F}^{(t)} =& \lambda \left\|\tilde{\mathbf{X}}_1^{(t-1)} - \mathbf{U}_1^{(t)}\tilde{\mathbf{V}}^{(t-1)}\right\|_F^2 + (1-\lambda)\left\|\tilde{\mathbf{X}}_2^{(t-1)} - \mathbf{U}_2^{(t)}\tilde{\mathbf{V}}^{(t-1)}\right\|_F^2 \\ &+ \lambda \left\|\mathbf{X}_1^{(t)} - \mathbf{U}_1^{(t)}\mathbf{V}^{(t)}\right\|_F^2 + (1-\lambda)\left\|\mathbf{X}_2^{(t)} - \mathbf{U}_2^{(t)}\mathbf{V}^{(t)}\right\|_F^2 \\ &+ \mu\left(\left\|\tilde{\mathbf{V}}^{(t-1)} - \mathbf{P}_1^{(t)}\tilde{\mathbf{X}}_1^{(t-1)}\right\| + \left\|\tilde{\mathbf{V}}^{(t-1)} - \mathbf{P}_2^{(t)}\tilde{\mathbf{X}}_2^{(t-1)}\right\|\right) \\ &+ \mu\left(\left\|\mathbf{V}^{(t)} - \mathbf{P}_1^{(t)}\mathbf{X}_1^{(t)}\right\| + \left\|\mathbf{V}^{(t)} - \mathbf{P}_2^{(t)}\mathbf{X}_2^{(t)}\right\|\right) \\ &+ \gamma R(\mathbf{U}_1^{(t)}, \mathbf{U}_2^{(t)}, \mathbf{P}_1^{(t)}, \mathbf{P}_2^{(t)}, \mathbf{V}^{(t)}, \tilde{\mathbf{V}}^{(t-1)}),\end{aligned} \tag{8}$$

where $\tilde{\mathbf{V}}^{(t-1)}$ and $\mathbf{V}^{(t)}$ are latent representation matrices for old and new data chunks, respectively. $\mathbf{U}_1^{(t)}, \mathbf{U}_2^{(t)}, \mathbf{P}_1^{(t)}, \mathbf{P}_2^{(t)}$ are matrix variables at round t.

It can be observed from (8) that the new data chunk $\mathbf{X}^{(t)}$ would not be able to significantly affect the optimization of latent representation matrix of old data $\tilde{\mathbf{V}}^{(t-1)}$, which mainly depends on the old data chunk $\tilde{\mathbf{X}}^{(t-1)}$ and parameters $\mathbf{U}_1^{(t)}, \mathbf{U}_2^{(t)}, \mathbf{P}_1^{(t)}, \mathbf{P}_2^{(t)}$. Therefore, in the online learning process, we mainly focus on updating matrix parameters $\mathbf{U}_1^{(t)}, \mathbf{U}_2^{(t)}, \mathbf{P}_1^{(t)}, \mathbf{P}_2^{(t)}$ and $\mathbf{V}^{(t)}$ by newly arriving data chunk $\mathbf{X}^{(t)}$.

The objective function $\mathcal{F}^{(t)}$ is convex with respect to any one of five matrix variables when others are fixed. Therefore, we propose an iterative algorithm to solve $\mathcal{F}^{(t)}$ as follows.

(1) Updating $\mathbf{U}_1^{(t)}$. By setting the partial derivative of $\mathcal{F}^{(t)}$ with respect to $\mathbf{U}_1^{(t)}$ to zero, we can update $\mathbf{U}_1^{(t)}$ by:

$$\mathbf{U}_1^{(t)} = \mathbf{E}_1^{(t)}\left(\mathbf{C}^{(t)} + \frac{\gamma}{\lambda}\mathbf{I}\right)^{-1}, \tag{9}$$

$$\mathbf{E}_1^{(t)} = \mathbf{X}_1^{(t)}\mathbf{V}^{(t)T} + \tilde{\mathbf{X}}_1^{(t-1)}\tilde{\mathbf{V}}^{(t-1)T} = \mathbf{X}_1^{(t)}\mathbf{V}^{(t)T} + \mathbf{E}_1^{(t-1)}, \tag{10}$$

$$\mathbf{C}^{(t)} = \mathbf{V}^{(t)}\mathbf{V}^{(t)T} + \tilde{\mathbf{V}}^{(t-1)}\tilde{\mathbf{V}}^{(t-1)T} = \mathbf{V}^{(t)}\mathbf{V}^{(t)T} + \mathbf{C}^{(t-1)}, \tag{11}$$

where $\mathbf{I}$ is an identity matrix. It can be easily observed that $\mathbf{E}_1^{(t-1)}$ and $\mathbf{C}^{(t-1)}$ are irrelevant to the new data chunk. They can be computed at previous round. Only $\mathbf{X}_1^{(t)}\mathbf{V}^{(t)T}$ and $\mathbf{V}^{(t)}\mathbf{V}^{(t)T}$ need to be calculated at current round. Therefore, $\mathbf{E}_1^{(t)}$ and $\mathbf{C}_1^{(t)}$ can be calculated incrementally with low storage and computation cost.

(2) Updating $\mathbf{U}_2^{(t)}$. Similar to the updating rule of $\mathbf{U}_1^{(t)}$, $\mathbf{U}_2^{(t)}$ can be updated by:

$$\mathbf{U}_2^{(t)} = \mathbf{E}_2^{(t)}\left(\mathbf{C}^{(t)} + \frac{\gamma}{1-\lambda}\mathbf{I}\right)^{-1}, \tag{12}$$

$$\mathbf{E}_2^{(t)} = \mathbf{X}_2^{(t)}\mathbf{V}^{(t)T} + \tilde{\mathbf{X}}_2^{(t-1)}\tilde{\mathbf{V}}^{(t-1)T} = \mathbf{X}_2^{(t)}\mathbf{V}^{(t)T} + \mathbf{E}_2^{(t-1)}. \tag{13}$$

(3) Updating $\mathbf{P}_1^{(t)}$. By setting the partial derivative of $\mathcal{F}^{(t)}$ with respect to $\mathbf{P}_1^{(t)}$ to zero, we can update $\mathbf{P}_1^{(t)}$ by:

$$\mathbf{P}_1^{(t)} = \mathbf{F}_1^{(t)}\left(\mathbf{W}_1^{(t)} + \frac{\gamma}{\mu}\mathbf{I}\right)^{-1}, \tag{14}$$

$$\mathbf{F}_1^{(t)} = \mathbf{V}^{(t)}\mathbf{X}_1^{(t)T} + \tilde{\mathbf{V}}^{(t-1)}\tilde{\mathbf{X}}_1^{(t-1)T} = \mathbf{V}^{(t)}\mathbf{X}_1^{(t)T} + \mathbf{F}_1^{(t-1)}, \tag{15}$$

$$\mathbf{W}_1^{(t)} = \mathbf{X}_1^{(t)}\mathbf{X}_1^{(t)T} + \tilde{\mathbf{X}}_1^{(t-1)}\tilde{\mathbf{X}}_1^{(t-1)T} = \mathbf{X}_1^{(t)}\mathbf{X}_1^{(t)T} + \mathbf{W}_1^{(t-1)}. \tag{16}$$

Similar to $\mathbf{E}_1^{(t)}$ and $\mathbf{C}^{(t)}$, $\mathbf{F}_1^{(t)}$ and $\mathbf{W}_1^{(t)}$ can also be calculated incrementally with low storage and computation cost.

(4) Updating $\mathbf{P}_2^{(t)}$. Similar to updating rule of $\mathbf{P}_1^{(t)}$, $\mathbf{P}_2^{(t)}$ can be updated by:

$$\mathbf{P}_2^{(t)} = \mathbf{F}_2^{(t)}\left(\mathbf{W}_2^{(t)} + \frac{\gamma}{\mu}\mathbf{I}\right)^{-1}, \tag{17}$$

$$\mathbf{F}_2^{(t)} = \mathbf{V}^{(t)}\mathbf{X}_2^{(t)T} + \tilde{\mathbf{V}}^{(t-1)}\tilde{\mathbf{X}}_2^{(t-1)T} = \mathbf{V}^{(t)}\mathbf{X}_2^{(t)T} + \mathbf{F}_2^{(t-1)}, \tag{18}$$

$$\mathbf{W}_2^{(t)} = \mathbf{X}_2^{(t)}\mathbf{X}_2^{(t)T} + \tilde{\mathbf{X}}_2^{(t-1)}\tilde{\mathbf{X}}_2^{(t-1)T} = \mathbf{X}_2^{(t)}\mathbf{X}_2^{(t)T} + \mathbf{W}_2^{(t-1)}. \tag{19}$$

(5) Updating $\mathbf{V}^{(t)}$. Ignoring irrelevant variable $\tilde{\mathbf{V}}^{(t-1)}$ in (8), we need to solve the following problem to obtain $\mathbf{V}^{(t)}$:

$$\begin{aligned}\min_{\mathbf{V}^{(t)}} & \lambda\left\|\mathbf{X}_1^{(t)} - \mathbf{U}_1^{(t)}\mathbf{V}^{(t)}\right\|_F^2 + (1-\lambda)\left\|\mathbf{X}_2^{(t)} - \mathbf{U}_2^{(t)}\mathbf{V}^{(t)}\right\|_F^2 \\ &+\mu\left(\left\|\mathbf{V}^{(t)} - \mathbf{P}_1^{(t)}\mathbf{X}_1^{(t)}\right\| + \left\|\mathbf{V}^{(t)} - \mathbf{P}_2^{(t)}\mathbf{X}_2^{(t)}\right\|\right) + \gamma R(\mathbf{V}^{(t)}).\end{aligned} \tag{20}$$

By setting the partial derivative of (20) with respect to $\mathbf{V}^{(t)}$ to zero, we can update $\mathbf{V}^{(t)}$ by:

$$\mathbf{V}^{(t)} = \left(\sum_{m=1}^{2}\lambda_m\mathbf{U}_m^{(t)T}\mathbf{U}_m^{(t)} + (2\mu+\gamma)\mathbf{I}\right)^{-1}\left(\sum_{m=1}^{2}\left(\lambda_m\mathbf{U}_m^{(t)T} + \mu\mathbf{P}_m^{(t)}\right)\mathbf{X}_m^{(t)}\right), \tag{21}$$

where $\lambda_1 = \lambda$ and $\lambda_2 = 1-\lambda$.

To initialize the aforementioned updating rules at each round t, we first initialize $\mathbf{V}^{(t)}$ by the corresponding matrix variables in the last round as

$$\mathbf{V}^{(t)} = \left(\sum_{m=1}^{2}\lambda_m\mathbf{U}_m^{(t-1)T}\mathbf{U}_m^{(t-1)} + (2\mu+\gamma)\mathbf{I}\right)^{-1}\left(\sum_{m=1}^{2}\left(\lambda_m\mathbf{U}_m^{(t-1)T} + \mu\mathbf{P}_m^{(t-1)}\right)\mathbf{X}_m^{(t)}\right). \tag{22}$$

Then, $\mathbf{V}^{(t)}$ can be used to calculate other matrix variables.

Algorithm 2 presents the whole optimization procedure at each round t ($t \geq 2$), where T denotes the number of iterations. Generally, small T is sufficient. We set $T = 5$ in the following experiments.

Regarding the convergence of iterations in Algorithm 2, we present the following theorem.

Theorem 1 *The objective function $\mathcal{F}^{(t)}$ in (8) is bounded. Algorithm 2 monotonically decreases the value of $\mathcal{F}^{(t)}$ in each iteration.*

Proof: As objective function $\mathcal{F}^{(t)}$ is a summation of norms with all positive penalty parameters, it is bounded as $\mathcal{F}^{(t)} \geq 0$. In Algorithm 2, the objective function $\mathcal{F}^{(t)}$ is divided into five subproblems. Each subproblem is a convex problem with respect to one variable. Therefore, variables $\mathbf{U}_1^{(t)}, \mathbf{U}_2^{(t)}, \mathbf{P}_1^{(t)}, \mathbf{P}_2^{(t)}$, and $\mathbf{V}^{(t)}$ obtained via updating rules (9), (12), (14), (17), and (21) are the exact minimum

points of subproblems. Consequently, the value of $\mathcal{F}^{(t)}$ is decreased in each iteration of Algorithm 2.

Based on Theorem 1, Algorithm 2 will converge to a local optimum as every bounded monotone sequence converges.

Algorithm 2 presents the updating process after the first round. At the first round, we need to learn all the matrix variables with the first data chunk instead of updating them. Actually, the objective function of OCMFH at the first round is equal to that of conventional CMFH. Therefore, we use CMFH method in Algorithm 1 to learn all the corresponding matrix variables of OCMFH at the first round.

Algorithm 2 Online optimizing algorithm at round t ($t \geq 2$)

Input: $\mathbf{X}_m^{(t)}, \mathbf{E}_m^{(t-1)}, \mathbf{F}_m^{(t-1)}, \mathbf{W}_m^{(t-1)}, \mathbf{U}_m^{(t-1)}, \mathbf{P}_m^{(t-1)}, \mathbf{C}^{(t-1)}, T$. ($m = 1, 2$)

Output: $\mathbf{E}_m^{(t)}, \mathbf{F}_m^{(t)}, \mathbf{W}_m^{(t)}, \mathbf{U}_m^{(t)}, \mathbf{P}_m^{(t)}, \mathbf{C}^{(t)}, \mathbf{V}^{(t)}$. ($m = 1, 2$)

Procedure:

1. Initialize $\mathbf{V}^{(t)}$ by (22).
2. Calculate $\mathbf{W}_1^{(t)}$ and $\mathbf{W}_2^{(t)}$.

for $iter \leq T$ **do**

3.1 Update $\mathbf{E}_1^{(t)}, \mathbf{E}_2^{(t)}, \mathbf{F}_1^{(t)}, \mathbf{F}_2^{(t)}, \mathbf{C}^{(t)}$.

3.2 Update $\mathbf{U}_1^{(t)}$ by (9).

3.3 Update $\mathbf{U}_2^{(t)}$ by (12).

3.4 Update $\mathbf{P}_1^{(t)}$ by (14).

3.5 Update $\mathbf{P}_2^{(t)}$ by (17).

3.6 Update $\mathbf{V}^{(t)}$ by (21).

end for

3.5 Hash Code Updating

When a new data chunk $\mathbf{X}^{(t)}$ arrives at round t, we can use Algorithm 2 to update parameters of hash model and generate unified representations $\mathbf{V}^{(t)}$. Whereas, unified representations $\tilde{\mathbf{V}}^{(t-1)}$ of old data chunk $\tilde{\mathbf{X}}^{(t-1)}$ are generated by the previous hash model learned at round $t-1$. As a result, there may be a mismatch between $\mathbf{V}^{(t)}$ and $\tilde{\mathbf{V}}^{(t-1)}$ because hash model has changed. Therefore, unified representations $\tilde{\mathbf{V}}^{(t-1)}$ of old data chunk should be updated according to model changes. The trivial solution is regenerating $\tilde{\mathbf{V}}^{(t-1)}$ by the new model. However, in such way, we have to access to $\tilde{\mathbf{X}}^{(t-1)}$ again in the round t, which obviously cannot meet the requirement of online learning.

In this paper, we propose a hash code updating strategy to update hash codes of old data to adapt to model changes without accessing to previous data points. Note that the goal of matrix factorization is to find two matrices whose product approximates the original data matrix as closely as possible. Therefore, the products of basis matrix and unified representations should approximate the original data matrices as closely as possible. Accordingly, let $\tilde{\mathbf{V}}_{new}^{(t-1)}$ denote the unified representations of old data generated by the updated model at round t, we have

$$\tilde{\mathbf{X}}_1^{(t-1)} \approx \mathbf{U}_1^{(t)} \tilde{\mathbf{V}}_{new}^{(t-1)} \approx \mathbf{U}_1^{(t-1)} \tilde{\mathbf{V}}^{(t-1)}. \tag{23}$$

Algorithm 3 Online collective matrix factorization hashing

Training stage

Input: Streaming data chunks $\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, \cdots, \mathbf{X}^{(s)}$, hash code length k, parameters λ, μ and γ, iteration number T.

Output: Hash codes $\mathbf{B}$, image and text projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$, image and text mean vectors $\mathbf{u}_1$ and $\mathbf{u}_2$.

Procedure:

1. Calculate means of $\mathbf{X}_1^{(1)}, \mathbf{X}_2^{(1)}$ and centering $\mathbf{X}_1^{(1)}, \mathbf{X}_2^{(1)}$.
2. Learn $\mathbf{E}_m^{(1)}, \mathbf{F}_m^{(1)}, \mathbf{W}_m^{(1)}, \mathbf{U}_m^{(1)}, \mathbf{P}_m^{(1)}, \mathbf{C}^{(1)}, \mathbf{V}^{(1)}$ with Algorithm 1. ($m = 1, 2$)
3. Set $\tilde{\mathbf{V}}^{(1)} = \mathbf{V}^{(1)}$.

for $t = 2 : s$ **do**

4.1 Calculate means of $\mathbf{X}_1^{(t)}, \mathbf{X}_2^{(t)}$ and update $\mathbf{u}_1$ and $\mathbf{u}_2$ by (4).

4.2 Centering $\mathbf{X}_1^{(t)}, \mathbf{X}_2^{(t)}$ by $\mathbf{u}_1, \mathbf{u}_2$.

4.3 Learn $\mathbf{E}_m^{(t)}, \mathbf{F}_m^{(t)}, \mathbf{W}_m^{(t)}, \mathbf{U}_m^{(t)}, \mathbf{P}_m^{(t)}, \mathbf{C}^{(t)}, \mathbf{V}^{(t)}$ with Algorithm 2.

4.4 Learn $\tilde{\mathbf{V}}_{new}^{(t-1)}$ by (26).

4.5 Set $\tilde{\mathbf{V}}^{(t)} = [\tilde{\mathbf{V}}_{new}^{(t-1)}, \mathbf{V}^{(t)}]$.

end for

5. Set $\mathbf{P}_1 = \mathbf{P}_1^{(s)}$, Set $\mathbf{P}_2 = \mathbf{P}_2^{(s)}$.
6. Calculate hash codes by $\mathbf{B} = \mathrm{sgn}(\tilde{\mathbf{V}}^{(s)})$.

Querying stage

Input: Feature vector $\mathbf{x}_1$ or $\mathbf{x}_2$, projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$, mean vectors $\mathbf{u}_1$ and $\mathbf{u}_2$.

Output: Hash code $\mathbf{b}$.

Procedure:

For $\mathbf{x}_1$:

1. Centering $\mathbf{x}_1$ by $\mathbf{u}_1$.
2. Calculate hash code by $\mathbf{b} = \mathrm{sgn}(\mathbf{P}_1 \mathbf{x}_1)$.

For $\mathbf{x}_2$:

1. Centering $\mathbf{x}_2$ by $\mathbf{u}_2$.
2. Calculate hash code by $\mathbf{b} = \mathrm{sgn}(\mathbf{P}_2 \mathbf{x}_2)$.

$$\tilde{\mathbf{X}}_2^{(t-1)} \approx \mathbf{U}_2^{(t)} \tilde{\mathbf{V}}_{new}^{(t-1)} \approx \mathbf{U}_2^{(t-1)} \tilde{\mathbf{V}}^{(t-1)}. \tag{24}$$

According to (23) and (24), the unified representations of old data can be updated by solving the following problem

$$\min_{\tilde{\mathbf{V}}_{new}^{(t-1)}} \mathcal{L}^{(t)} = \sum_{m=1}^{2} \lambda_m \left\| \mathbf{U}_m^{(t)} \tilde{\mathbf{V}}_{new}^{(t-1)} - \mathbf{U}_m^{(t-1)} \tilde{\mathbf{V}}^{(t-1)} \right\|_F^2 + \gamma R(\tilde{\mathbf{V}}_{new}^{(t-1)}), \tag{25}$$

where $\lambda_1 = \lambda$ and $\lambda_2 = 1 - \lambda$ control the weights of image and text. By setting the partial derivative of $\mathcal{L}^{(t)}$ with respect to $\tilde{\mathbf{V}}_{new}^{(t-1)}$ to zero, we can obtain

$$\tilde{\mathbf{V}}_{new}^{(t-1)} = \left(\sum_{m=1}^{2} \lambda_m \mathbf{U}_m^{(t)T} \mathbf{U}_m^{(t)} + \gamma \mathbf{I} \right)^{-1} \left(\sum_{m=1}^{2} \lambda_m \mathbf{U}_m^{(t)T} \mathbf{U}_m^{(t-1)} \tilde{\mathbf{V}}^{(t-1)} \right) \tag{26}$$

Therefore, unified representations of old data can be efficiently updated to adapt to hash model changes through (26) without accessing to old data points. As hash codes are obtained by quantizing

unified representations, they have also been updated with the updating of unified representations.

The whole learning procedure of OCMFH is summarized in Algorithm 3.

3.6 Complexity Analysis

In this section, we analyze the time and space complexity of the proposed OCMFH. As OCMFH incrementally learns hash functions and hash codes in a streaming manner, we analyze the time and space complexity of OCMFH at each round t.

Time Complexity: The time complexity of OCMFH is mainly determined by online optimization procedure. In online optimization process, there are five matrix variables need to be updated. The complexity of updating each variable is $O(n_t)$, n_t is the size of data chunk at round t. As the number of iterations is usually very small, the overall time complexity of online optimization process is $O(n_t)$, which is linear to the size of new data chunk. Therefore, the proposed OCMFH is efficient in time complexity.

Space Complexity: At each round, OCMFH preserves intermediate matrices for the updating at the next round. The sizes of these matrices are only related to hash code length and feature vectors, and they occupy not much memory space. Of all the variables OCMFH used in the learning process, only sizes of data matrices and unified representation matrix are related to n_t. In general, the overall space complexity of OCMFH is $O(n_t)$, which is efficient in large-scale retrieval task.

3.7 Multiple Modalities Extension

The objective function of OCMFH for multimodal data is the same with (1). We can adapt Algorithm 3 to online optimize problem (1).

4 EXPERIMENTS

In this section, we compare the proposed OCMFH with several state-of-the-art online cross-modal hashing methods on three benchmark data sets. The most commonly used metric mean of average precision (mAP) is adopted to evaluate the cross-media retrieval performance [23]. mAP averages mean precision values of top-k results over all the queries. In the experiments, k is set to 100. Firstly, we introduce data sets, comparison methods, and experimental settings that are related to the following experiments. Then, we report and analyze the experimental results. Finally, ablation analysis, convergence analysis, computational efficiency and parameter sensitivity of OCMFH are further investigated.

4.1 Data Sets

Three multimodal data sets are used to evaluate the performance of OCMFH. Each data set has two modalities, i.e. image and text. For each query data, its semantic neighbors are defined as those sharing at least one semantic label with it. Statistics of these three data sets are introduced as following.

MIRFlickr [8] data set consists of 25,000 image-text data pairs collected from Flickr website. Each data pair is associated with one or more of the 24 semantic labels. Following settings in literature [12], we obtain 16,738 pairs. Each image is represented as a 150-dimensional edge histogram feature and each text is represented as a 500-dimensional feature by performing PCA on the index vector. We randomly select 836 image-text pairs to serve as the query set and the remaining pairs are served as the training set. To support online learning, the training set is split to 8 data chunks, with the first 7 chunks contain 2,000 data pairs each and the last chunk contains 1902 pairs.

NUS-WIDE [4] data set consists of 269,648 image-text data pairs collected from Flickr website. Each data pair is associated with one or more of the 81 semantic labels. We select a subset consists of 186,577 data pairs which are corresponding to the top 10 most frequent labels following literature [24]. Each image is represented as a 128-dimensional feature by performing PCA on its 4,096-dimensional deep feature extracted by the Caffe implementation of VGG Net [21], and each text is represented as a 1,000-dimensional bag-of-words feature. We randomly select 2,000 image-text pairs to serve as the query set and the remaining pairs are served as the training set. To support online learning, the training set is split to 37 data chunks, with the first 36 chunks contain 5,000 data pairs each and the last chunk contains 4,577 pairs.

MSCOCO [11] data set consists of 122,218 labeled image-text data pairs collected from Flickr website. Each data pair is associated with one or more of the 80 semantic labels. Each image is represented as a 512-dimensional feature by performing PCA on its 4,096-dimensional deep feature extracted by the Caffe implementation of VGG Net, and each text is represented as a 512-dimensional feature by performing PCA on its index vector. We randomly select 2,000 image-text pairs to serve as the query set and the remaining pairs are served as the training set. To support online learning, the training set is split to 25 data chunks, with the first 24 chunks contain 5,000 data pairs each and the last chunk contains 218 pairs.

4.2 Baselines and Experimental Settings

To the best of our knowledge, there are two unsupervised online cross-modal hashing methods, i.e. OCMH [27] and DMVH [26]. Therefore, we compare the proposed OCMFH with them. In consideration of OCMFH is an online version of CMFH, we also compare OCMFH with CMFH. As CMFH is a batch-based method, we use all the training data points to train it in only one round. To show the effectiveness of incrementally update hash functions by OCMFH, we also use the first data chunk to train hash functions by CMFH and then apply the learned hash functions to generate hash codes for other training data chunks, and let CMFHBatch denote the performance obtained in this way.

OCMFH and CMFH have three parameters: λ, μ, and γ. We set $\lambda = 0.5$, $\mu = 100$, and $\gamma = 0.001$ in the following experiments. The maximum number of iterations at each round for OCMFH is set to 5. CMFH trains all the training data in one round and the maximum number of iterations is set to 100. Parameters of OCMH and DMVH are tuned in terms of their literatures. Note that the code length of DMVH augments continuously. In the experiments, we set its initial encoding length to the fixed hash code length of other methods.

4.3 Experimental Results

The mAP values of OCMFH and all the comparison methods with different hash code lengths on three data sets are reported in Figure 3. From it, we have the following observations.

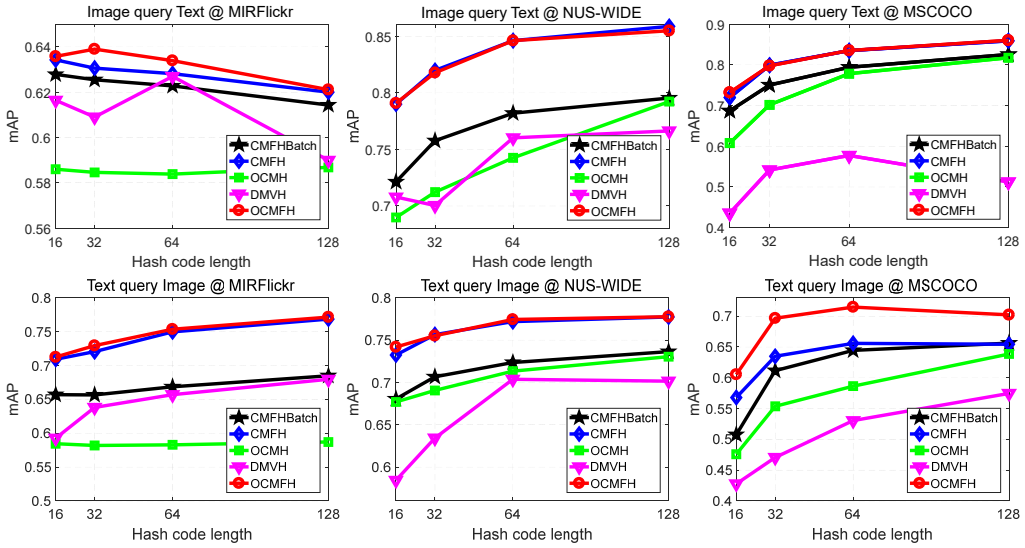

Figure 3: mAP comparisons on different data sets with different hash code lengths.

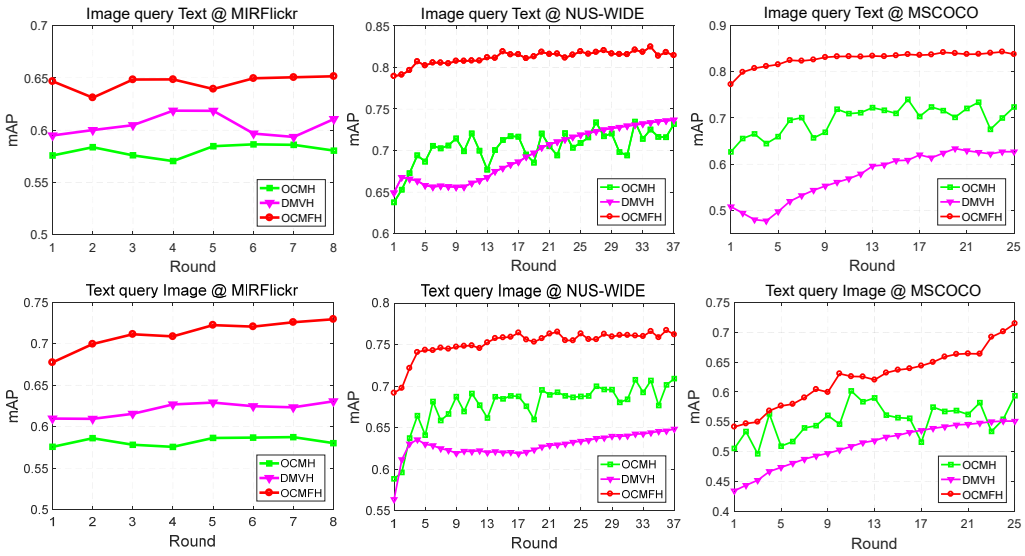

Figure 4: mAP comparisons on different data sets at each round with 32 bits hash code length.

Table 1: mAP comparisons of OCMFH with and without hash code updating scheme.

Task	Method	MIRFlickr			NUS-WIDE			MSCOCO		
		32 bits	64bits	128bits	32 bits	64bits	128bits	32 bits	64bits	128bits
Image query Text	OCMFH1	0.6403	0.6262	0.6143	0.8188	0.8419	0.8546	0.7998	0.8406	0.8590
	OCMFH	**0.6415**	**0.6358**	**0.6226**	**0.8207**	**0.8455**	**0.8568**	**0.8046**	**0.8448**	**0.8662**
Text query Image	OCMFH1	0.7261	0.7497	0.7719	0.7674	0.7782	0.7842	0.5661	0.5726	0.6013
	OCMFH	**0.7264**	**0.7508**	**0.7732**	**0.7675**	**0.7783**	**0.7863**	**0.7041**	**0.6889**	**0.6582**

1) Intuitively, the proposed OCMFH achieves the highest mAP values compared with online hashing methods OCMH and DMVH on all the data sets in terms of both image-query-text and text-query-image tasks. This shows the effectiveness of OCMFH in online cross-media retrieval.

2) OCMFH yields better performance than CMFHBatch. It reflects that OCMFH can update hash functions and hash codes according to the variations of streaming data. With hash model continuously updating, the performance of OCMFH can be consistently improved.

3) OCMFH leads to comparable results as CMFH. In some cases, OCMFH even achieves slightly higher mAP values than CMFH. This phenomenon suggests that OCMFH can incrementally learn hash model of CMFH in a streaming manner with competitive performance.

In summary, experimental results in Figure 3 illustrate that, by dynamically updating the hash model with variations of streaming data, OCMFH can achieve promising retrieval performance.

Figure 4 shows the mAP values of all the online hashing methods on three data sets at each round with 32 bits hash code length. It can be observed that generally mAP values of each method are increased with the increase of available training data points. This reflects that online methods can make the model more fit for data along with the increasing of rounds. Compared with OCMF and DMVH, the proposed OCMFH significantly performs better at each round on all the data sets, which indicates its superior performance.

4.4 Ablation Analysis

The proposed OCMFH can dynamically update hash codes of old data with the changes of hash model without referring to original old data. To evaluate the effectiveness of this hash code updating scheme, we design a variant of OCMFH named OCMFH1 which dose not update hash codes of old data. Table 1 reports mAP values of OCMFH and OCMFH1 on three data sets with different code lengths. It is obvious that OCMFH always performs better than OCMFH1. This suggests that updating hash codes can better match hash codes of old and new data, and thus improve the retrieval performance.

4.5 Convergence Analysis

In this section, we experimentally show the convergence property of OCMFH. In the experiment, hash code length is set to 32 bits and the iteration number of each round is set to 5. Figure 5 shows the convergence curves of OCMFH with the arrival of streaming data on three data sets. It can be observed that the objective function value is decreased monotonously at each step in each round. And the objective function value generally presents a declining trend with the increase of rounds. As the number of iterations increases, the objective function value will eventually reach a local minimum.

4.6 Training Time Analysis

Figure 6 shows the training time of each method by varying the size of the training set on three data sets with 32 bits hash code length. It can be observed that OCMFH costs sligtly more time than CMFH-Batch but takes much less time than CMFH. As CMFHBatch trains hash model with only the first data chunk, of course, it needs less training time than OCMFH and CMFH which train hash model with all the training data. Compared with CMFH, OCMFH consumes less time with the same training size. This reflects the computation efficiency of online learning. Of all three online hashing methods, OCMFH is much faster than DMVH and slightly slower than OCMH. When training size is greater than 14,000, online hashing method

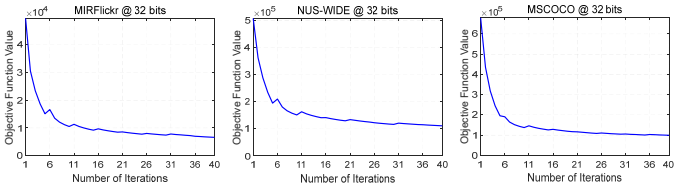

Figure 5: Convergence curves of OCMFH.

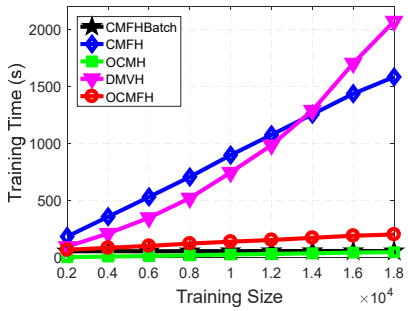

Figure 6: Training time (in seconds) comparisons on NUS-WIDE with different training sizes.

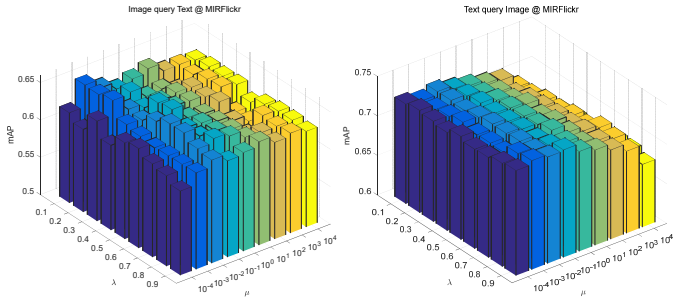

Figure 7: mAP variations with parameters λ and μ with 32 bits hash code length on MIRFlickr.

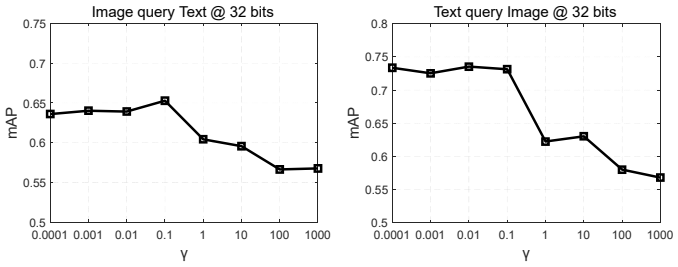

Figure 8: mAP variations with parameter γ with 32 bits hash code length on MIRFlickr.

DMVH even takes more time than batch-based method CMFH. According to these observations, the proposed CMFH has a competitive training efficiency as well as better performance compared with existing online cross-modal hashing methods.

4.7 Parameter Sensitivity Analysis

In this section, we empirical analyze the sensitivity of each parameter of OCMFH on MIRFlickr. The hash code length is set to 32. When analyzing the specific parameters, we fix other parameters and vary their values. Figure 7 shows the mAP variations with λ and μ. Usually, OCMFH achieves well performance when μ within the range of $[0.1, 100]$. λ exerts very little influence on OCMFH on text query image task. On image query text task, OCMFH performs well when the value of λ is around 0.5. Generally, λ can be chosen within the range of $[0.3, 0.7]$. Figure 8 shows the mAP variations with parameter γ. It can be observed that the stable performance of OCMFH can be achieved when γ is less than 0.1. Generally, γ can be chosen within the range of $[0.0001, 0.1]$.

5 CONCLUSION

In this paper, we propose an effective and efficient online cross-modal hashing method named online collective matrix factorization based on batch-based collective matrix factorization hashing. It incrementally updates hash functions and generates hash codes by newly arriving data. Through processing one data chunk at a time, it achieves great savings in computation and storage. Moreover, it adaptively updates hash codes of old data with variations of hash model without accessing to old data. In such way, hash codes of old data and new data are better matched and hence the retrieval performance is enhanced. Extensive experiments on three benchmark data sets have demonstrated that the proposed method outperforms several state-of-the-art methods in terms of both accuracy and efficiency. In the future work, we will try to extend the proposed method to a deep matrix factorization model to boost its performance.

6 ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grants 61702394, 61772402, 61972302, 61876146, 61802294 and 61702400, in part by the China Postdoctoral Science Foundation funded project under Grants 2018T111021 and 2017M613082, in part by the National Key Research and Development Program of China under Grant 2016QY01W0200, in part by the Scientific Research Project of Shaanxi Province, China under Grants 2019JQ-205, 2019ZDLGY13-01, 2019ZDLGY13-07 and 2017ZDXM-GY-002, and in part by the Science and Technology Project of Xi'an, China under Grant 201809170CX11JC12.

REFERENCES

[1] Fatih Cakir, Sarah Adel Bargal, and Stan Sclaroff. 2017. Online supervised hashing. *Computer Vision and Image Understanding* 156 (2017), 162–173.
[2] Fatih Cakir, Kun He, Sarah Adel Bargal, and Stan Sclaroff. 2017. MIHash: online hashing with mutual information. In *Proceedings of the 16th International Conference on Computer Vision*. IEEE, 437–445.
[3] Xixian Chen, Haiqin Yang, Shenglin Zhao, Michael R Lyu, and Irwin King. 2019. Making online sketching hashing even faster. *IEEE Transactions on Knowledge and Data Engineering* (2019).
[4] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 2009. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the 8th ACM International Conference on Image and Video Retrieval*. ACM, 1–9.
[5] Guiguang Ding, Yuchen Guo, and Jile Zhou. 2014. Collective matrix factorization hashing for multimodal data. In *Proceedings of the 27th International Conference on Computer Vision and Pattern Recognition*. IEEE, 2075–2082.
[6] Di Hu, Feiping Nie, and Xuelong Li. 2018. Deep binary reconstruction for cross-modal hashing. *IEEE Transactions on Multimedia* 21, 4 (2018), 973–985.
[7] Long-Kai Huang, Qiang Yang, and Wei-Shi Zheng. 2013. Online hashing. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*. IJCAI, 1422–1428.
[8] Mark J Huiskes and Michael S Lew. 2008. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM International Conference on Multimedia Information Retrieval*. ACM, 39–43.
[9] Cong Leng, Jiaxiang Wu, Jian Cheng, Xiao Bai, and Hanqing Lu. 2015. Online sketching hashing. In *Proceedings of the 28th International Conference on Computer Vision and Pattern Recognition*. IEEE, 2503–2511.
[10] Mingbao Lin, Rongrong Ji, Hong Liu, and Yongjian Wu. 2018. Supervised online hashing via hadamard codebook learning. In *Proceedings of the 26th ACM International Conference on Multimedia*. ACM, 1635–1643.
[11] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: common objects in context. In *European Conference on Computer Vision*. Springer, 740–755.
[12] Zijia Lin, Guiguang Ding, Mingqing Hu, and Jianmin Wang. 2015. Semantics-preserving hashing for cross-view retrieval. In *Proceedings of the 28th International Conference on Computer Vision and Pattern Recognition*. IEEE, 3864–3872.
[13] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan, and Jie Zhou. 2017. Cross-modal deep variational hashing. In *Proceedings of the 16th International Conference on Computer Vision*. IEEE, 4097–4105.
[14] Li Liu, Zijia Lin, Ling Shao, Fumin Shen, Guiguang Ding, and Jungong Han. 2016. Sequential discrete hashing for scalable cross-modality similarity retrieval. *IEEE Transactions on Image Processing* 26, 1 (2016), 107–118.
[15] Xu Lu, Lei Zhu, Zhiyong Cheng, Jingjing Li, Xiushan Nie, and Huaxiang Zhang. 2019. Flexible online multi-modal hashing for large-scale multimedia retrieval. In *Proceedings of the 27th ACM International Conference on Multimedia*. ACM, 1129–1137.
[16] Xu Lu, Lei Zhu, Zhiyong Cheng, Liqiang Nie, and Huaxiang Zhang. 2019. Online multi-modal hashing with dynamic query-adaption. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 715–724.
[17] Wing WY Ng, Xing Tian, Yueming Lv, Daniel S Yeung, and Witold Pedrycz. 2016. Incremental hashing for semantic image retrieval in nonstationary environments. *IEEE Transactions on Cybernetics* 47, 11 (2016), 3814–3826.
[18] Wing WY Ng, Xing Tian, Witold Pedrycz, Xizhao Wang, and Daniel S Yeung. 2018. Incremental hash-bit learning for semantic image retrieval in nonstationary environments. *IEEE Transactions on Cybernetics* 49, 11 (2018), 3844–3858.
[19] Yuxin Peng, Xin Huang, and Yunzhen Zhao. 2018. An overview of cross-media retrieval: concepts, methodologies, benchmarks, and challenges. *IEEE Transactions on Circuits and Systems for Video Technology* 28, 9 (2018), 2372–2385.
[20] Dimitrios Rafailidis and Fabio Crestani. 2016. Cluster-based joint matrix factorization hashing for cross-modal retrieval. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 781–784.
[21] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556* (2014).
[22] Ajit Singh and Geoffrey Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 650–658.
[23] Jingkuan Song, Yang Yang, Yi Yang, Zi Huang, and Heng Tao Shen. 2013. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *Proceedings of 19th ACM SIGMOD International Conference on Management of Data*. ACM, 785–796.
[24] Di Wang, Xinbo Gao, Xiumei Wang, and Lihuo He. 2019. Label consistent matrix factorization hashing for large-Scale cross-modal similarity search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 41, 10 (2019), 2466–2479.
[25] Zhenyu Weng and Yuesheng Zhu. 2019. Online supervised sketching hashing for large-Scale image retrieval. *IEEE Access* 7 (2019), 88369–88379.
[26] Liang Xie, Jialie Shen, Jungong Han, Lei Zhu, and Ling Shao. 2017. Dynamic multi-view hashing for online image retrieval. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. IJCAI, 3133–3139.
[27] Liang Xie, Jialie Shen, and Lei Zhu. 2016. Online cross-modal hashing for web image retrieval. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. AAAI, 294–300.
[28] Tao Yao, Gang Wang, Lianshan Yan, Xiangwei Kong, Qingtang Su, Caiming Zhang, and Qi Tian. 2019. Online latent semantic hashing for cross-media retrieval. *Pattern Recognition* 89 (2019), 1–11.
[29] Dongqing Zhang and Wu-Jun Li. 2014. Large-scale supervised multimodal hashing with semantic correlation maximization. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*. AAAI, 2177–2183.
[30] Jile Zhou, Guiguang Ding, and Yuchen Guo. 2014. Latent semantic sparse hashing for cross-modal similarity search. In *Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 415–424.