View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Lancaster E-Prints

Dynamic Multi-View Hashing for Online Image Retrieval

Paper ID:888

Abstract

Advanced hashing technique is essential to facilitate effective organization and retrieval over online image collections, where the contents are frequently changed. Traditional multi-view hashing methods based on batch-based learning generally leads to very expensive updating cost. Meanwhile, existing online hashing methods mainly focus on single-view data and thus can not achieve promising performance when searching real online images, which are multiple view based data. In this paper, we propose dynamic multi-view hashing (DMVH), which can adaptively augment hash codes according to dynamic changes of image. Meanwhile, DMVH leverages online learning to generate hash codes. It can increase the code length when current code is not able to represent new images effectively. Moreover, to gain further improvement on overall performance, each view is assigned with a weight, which can be efficiently updated during the online learning process. In order to avoid the frequent updating of code length and view weights, an intelligent buffering scheme is also specifically designed to preserve significant data to maintain good effectiveness of DMVH. Experimental results on two real-world image datasets demonstrate superior performance of DWVH over several state-of-the-art hashing methods.

1 Introduction

With the explosive growth of online image repositories, how to develop intelligent hashing techniques to facilitate large scale image access has attracted more and more attentions in recent years. Despite different visual features, the Web images are often associated with text information (e.g., tags or short comments). In order to improve search performance, various kinds of multi-view hashing schemes have been proposed to integrate the competency of both hashing technology and multi-view learning [Zhang *et al.*, 2011; Wu *et al.*, 2014; 2015]. Comparing to single-view hashing, it not only enjoys better search performance but also can support the search queries from different views (e.g., those based on visual contents or textual information).

Online image collection is highly dynamic and frequently updated. Since traditional multi-view hashing methods are designed based on batch-based learning, its updating cost could be very expensive under online environment. Further, when new images arrive, multi-view hashing needs to accumulate all the data to retrain the hashing functions. Consequently it leads to extremely high time complexity, which is unaffordable. Motivated by the concerns, several online hashing methods [Leng *et al.*, 2015; Cakir and Sclaroff, 2015b] have been proposed to support effective search over dynamic image databases. However, their performance is still far beyond satisfactory. When applying existing multi-view hashing and online hashing methods to Web image retrieval, they generally suffer from several issues. Existing online hashing methods aim at generating the binary codes with fixed length, which can achieve optimal signature of the dynamic image data. Meanwhile, the contents of new image could be semantically different from the old one. When new image is inserted into the database, longer code is required to preserve more discriminative information to ensure accurate search over updated image collections. According to previous results [Zhou *et al.*, 2014], longer hash code can lead to more comprehensive image representation and thus better search performance. For example, Wikipedia dataset [Rasiwasia *et al.*, 2010] consisting of 2866 images only needs 32 bits codes to achieve the optimal accuracy. For NUS-WIDE having 269648 images, at least 128 bits are required to gain the best performance. However, larger code length does not mean the better performance, in that hashing for data with small size may get trapped in local minima [Zhen and Yeung, 2012]. Therefore, general hashing methods usually choose an appropriate code length for a particular database, which is not suitable to dynamic image collections. To solve this problem, the code length for dynamic image database needs to be intelligently augmented to guarantee hashing performance. On the other hand, it is crucial to weight importance of different views in the online learning of hash codes since different view gives different contributions to discriminative capability of the final hash codes. For example, text feature usually contains more semantics than visual feature does [Caicedo *et al.*, 2012], thus it should be assigned with the higher weight. While several multi-view hashing methods take the view weights into account, to the best of our knowledge, none of existing online hashing methods focus on developing intelligent scheme to estimate optimal weighs for different views.

In this paper, we propose a novel unsupervised online hashing method called Dynamic Multi-view Hashing (DMVH) to support the effective and efficient online image retrieval. Towards this goal, DMVH is developed based on a dynamic hashing scheme, where length of hashing codes can be adaptively augmented. In DMVH, a dictionary is constructed to infer the hash codes of images. When a new data cannot be represented by current dictionary, it will be augmented the dictionary. Moreover, DMVH applies multi-view features for hashing, and proper weight is estimated to each view. In the online learning process, to avoid the frequent updating of dictionary and modality weights, a buffering scheme is designed to preserve the images for purpose of subsequential updating. The core technical contributions of our work can be summarized as follows:

1. In DMVH, the code length can be dynamically augmented with the changes of image data. As a result, DMVH can better represent the streaming images. In addition, users are not required to predefine the appropriate code length which will be automatically obtained by DMVH.
2. DMVH can effectively estimate proper weights for different views to reflect their different importance in hashing. Meanwhile, an intelligent buffer scheme is designed for the online learning process, which enables highly efficient learning process of hash codes and view weights.
3. Experimental results on real-world image datasets demonstrate the superiority of DMVH in terms of both search efficiency and accuracy.

2 Related Work

Multi-view Hashing: Recent years have witnessed fast development of multi-view hashing techniques due to its wide range of real applications. Existing multi-view hashing methods can be generally divided to two main categories according to the query types supported. The multi-view hashing methods combine all features to construct database hash codes, but they require that query has the same features as database data. This type of multi-view hashing usually aims to gain the effective combination of multi-view features. Composite Hashing with Multiple Information Sources (CHMIS) [Zhang *et al.*, 2011] leverages graphs of multi-views features to learn the hash codes, and each view is assigned with a weight for graph combination. Since graph learning is inefficient, anchor graph is adopted to accelerate the modeling. Multi-view Anchor Graph Hashing (MVAGH) [Kim and Choi, 2013] exploits anchors to construct graphs to achieve more effective hashing. Multiview Alignment Hashing (MAH) [Liu *et al.*, 2015] combines matrix factorization with anchor graph learning for hashing, and it achieves good performance based on multi-view features. Multi-view hashing is also called as cross-view hashing since it can support image search based on queries with various view. Generally, cross-view hashing methods aim at modeling the correlation of different views. For example, as an extended version of Canonical Correlation Analysis (CCA), Cross-view Hashing (CVH) [Kumar and Udupa, 2011] optimizes the hamming distance of different views. Inter-media Hashing (IMH) [Song *et al.*, 2013] both optimizes the intra-media and inter-media consistency. Semantic Correlation Maximization (SCM) [Zhang and Li, 2014] maximizes the semantic correlation between image and text features, and it further considers the quantization loss of hash codes. Semantic Topic Multimodal Hashing (STMH) [Zhou *et al.*, 2014] firstly generates text topics and image concepts, and then combines them in a common hash space.

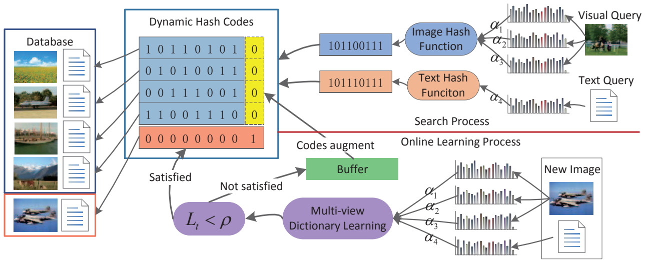


Figure 1: The overall illustration of DMVH.

Meanwhile, a few multi-view hashing methods were developed to both support any single-view query and combine multi-few features. Collective Matrix Factorization Hashing (CMFH) [Ding *et al.*, 2014] uses Collective Matrix Factorization (CMF) to combine image and text features, and it simultaneously learns the hash function of single image feature and text feature. Although current multi-view hashing methods perform well in combining or correlating multi-view features, they are not suited to streaming data where images are s sequentially inserted into the database. Moreover, processing multi-view features under online learning environment is more challenging than batch-based learning.

Online Hashing: Online hashing has been emerging as a promising technique to generate compact binary code under dynamic environment. Due to its practical value in real applications, numerous online hashing methods have been proposed recently and can be classified into two major categories: supervised learning based and unsupervised learning based. In particular, Online Kernel Hashing (OKH)[Huang *et al.*, 2013] and Adaptive Hashing [Cakir and Sclaroff, 2015a] are typical examples of supervised methods. Both of them use similar image pairs as training data. Although supervised learning based method can effectively preserve the correlation of images in hash codes, large amount of labeled training examples are required and could be very expensive to collect. On the other hand, Online Sketching Hashing (OSH) [Leng *et al.*, 2015] is unsupervised online hashing method, which doesn't rely on labeled data. However, the main disadvantage of existing online hashing methods is their incapability to achieve high quality hash code. Thus, good retrieval performance can not be promised.

3 Dynamic Multi-View Hashing

As illustrated in Figure 1, DMVH consists of two main modules: online learning process and search process. In the dynamic hash codes, blue area, red area and yellow area denote the old codes, hash codes of new data and the augmented codes respectively. The database includes images having both visual and text contents. When new image arrives, multiple kinds of visual features and one text feature are extracted and combined via multi-view dictionary learning. The differentiation threshold L_t determines whether the image can be represented using current hash codes. If not, it is added to the buffer, and the buffer data are used to augment hash codes till no more space is available in the buffer. In the following sections, more details about L_t will be presented.

3.1 Model Formulation

In DMVH, at step t, the database contains old image features added at previous steps $X_{t-1}^{m}|_{m=1}^{M}$, where M is the number of views. $X_{t-1}^{m} \in \mathbb{R}^{(t-1)\times d_m}$, and d_m is the dimension of m_{th} view feature. The old images are represented with hash codes $H_{t-1} \in \{0,1\}^{(t-1)\times C}$, where C is the code length. Then new image $x_t^m|_{m=1}^{M}$ is added into the database, and $X_t^m = [(X_{t-1}^m)^T, (x_t^m)^T]^T$. Our goal is to learn new hash codes $h_t \in \{0,1\}^{1\times C}$ to represent new image. The hash function also needs to be learnt to project any views into the learned hash space. Towards the goal, the kernel dictionary representation is gained to construct hash codes. Dictionary learning is effective in learning hash codes [Yu *et al.*, 2014; Zhang *et al.*,], and the kernel projection will make the hash codes preserve more discriminative information. Thus, at time t, the dictionary consists of a subset of the database samples with muti-view features. For each view, its corresponding dictionary is denoted as $D_t^m = [(x_{a_1}^m)^T, \dots, (x_{a_C}^m)^T]^T$. Assuming each sample can be constructed from its hash codes and dictionary, we have the following formulation:

$$\begin{aligned} min \quad & L_t\left(H_t, D_t^m|_{m=1}^{M}\right) = \\ & \sum_{i=1}^{M} \alpha_m^2 \left\| H_t \phi\left(D_t^m\right) - \phi\left(X_t^m\right) \right\|_F^2 + \lambda \left\| H_t \right\|_F^2 \\ s.t. \quad & \sum_{m=1}^{M} \alpha_m = 1 \end{aligned} \tag{1}$$

where $\phi(\cdot)$ is a kernel projection function, λ is the regularization parameter, $\alpha = [\alpha_1, \dots, \alpha_M]$ denotes the view weights.

3.2 Online Learning

Basic process At each step t, when new image is added into the database, the overall objective function can be reformulated as:

$$\begin{aligned} min \quad & L_t\left(H_t, D_t^m|_{m=1}^{M}\right) = \\ & L_{t-1}\left(H_{t-1}, D_t^m|_{m=1}^{M}\right) + l_t\left(h_t, D_t^m|_{m=1}^{M}\right) \end{aligned} \tag{2}$$

If dictionary $D_t^m|_{m=1}^{M}$ is not changed, $D_t^m = D_{t-1}^m$, the constraint $L_{t-1}\left(H_{t-1}, D_t^m|_{m=1}^{M}\right)$ has been satisfied by previous learning steps. Thus, We only need to consider the optimization of $l_t\left(h_t, D_t^m|_{m=1}^{M}\right)$, and the objective function is:

$$\begin{aligned} min \quad & l_t\left(h_t, D_t^m|_{m=1}^{M}\right) = \\ & \sum_{m=1}^{M} \alpha_m^2 \left\| h_t \phi\left(D_t^m\right) - \phi\left(x_t^m\right) \right\|_F^2 + \lambda \left\| h_t \right\|_F^2 \end{aligned} \tag{3}$$

After applying the kernel trick, the objective function (3) can be rewritten as:

$$\sum_{m=1}^{M} \alpha_m^2 \left(h_t \widetilde{K}_t^m h_t^T - 2h_t (z_t^m)^T + k_{tt}^m \right) + \lambda h_t h_t^T \tag{4}$$

where $\widetilde{K}_t^m \in \mathbb{R}^{C\times C}$ is the kernel matrix of the dictionary D_t^m, and $\widetilde{K}_t^m(i,j) = \phi(x_{a_i}^m)\phi(x_{a_j}^m)^T$, $(z_t^m)_i = \phi(x_t^m)\phi(x_{a_i}^m)^T$, a_i is the index of i_{th} dictionary element. $k_{tt}^m = \phi(x_t^m)\phi(x_t^m)^T$.

By setting the derivative of Eq.(4) to zeros, we can obtain h_t as:

$$h_t = z_t\left(\widetilde{K}_t + \lambda I\right)^{-1} \tag{5}$$

where $\widetilde{K}_t = \sum_{m=1}^{M} \alpha_m^2 \widetilde{K}_t^m, z_t = \sum_{m=1}^{M} \alpha_m^2 z_t^m$.

For each modality m, if vector $\phi\left(x_t^m\right)$ is linearly independent on vectors of dictionary $\phi\left(D_t^m\right)$. Then h_t computed by Eq.(5) cannot satisfy Eq.(3), and x_t^m should be added into the dictionary. Therefore, we have to determine the linear dependence of new vectors. By substituting Eq.(5) into Eq.(3), it can be transformed to:

$$l_t\left(h_t, D_{t-1}^m|_{m=1}^{M}\right) = \sum_{m=1}^{M} \alpha_m^2 \left(k_{tt}^m - z_t^m h_t\right) + \lambda h_t h_t^T \tag{6}$$

We use a threshold ρ to determine the linear dependence. If $l_t\left(h_t, D_{t-1}^m|_{m=1}^{M}\right) < \rho$, then the approximate linear dependence described by Eq.(3) can be satisfied. And we do not need to change dictionary. If $l_t\left(h_t, D_{t-1}^m|_{m=1}^{M}\right) \geq \rho$, then new hash codes cannot be effectively represented by current dictionary. As a result, both dictionary and hash codes need to be augmented, and the modal weights should be recomputed accordingly.

Dictionary Augment with Buffer Scheme Considering the worst case that new image is always very different from database images and cannot be represented by current dictionary, frequent dictionary updating is unavoidable and obviously inefficient. Therefore, a buffer scheme is applied to avoid the frequent updating process. The buffer scheme relaxes the restrict linear dependence in Eq.(3). At each step t, if the new image satisfies $l_t\left(h_t, D_{t-1}^m|_{m=1}^{M}\right) \geq \rho$, then image t is inserted into the buffer with maximum size B_{max}. After the updating, in case that the buffer is full, image b will be selected from the buffer with highest value of l_t. Then we augment the dictionary with image b and update the view weights α. The dictionary is updated with $D_t^m = [(D_{t-1}^m)^T, (x_b^m)^T]^T$, and the hash codes constructed from dictionary have to be augmented accordingly. We should optimize the following overall objective function:

$$\begin{aligned} min \quad & L_t\left(H_t, D_t^m|_{m=1}^{M}\right) = \\ & L_{t\backslash b}\left(H_{t\backslash b}, D_t^m|_{m=1}^{M}\right) + l_b\left(h_b, D_t^m|_{m=1}^{M}\right) \\ s.t. \quad & \sum_{m=1}^{M} \alpha_m = 1 \end{aligned} \tag{7}$$

Where $H_{t\backslash b}$ denotes the hash codes of database images except b. Since image b is linearly independent on other dictionary images, we can only use it to represent itself. Therefore we have $h_b = [\mathbf{0}, 1]$, and $l_b\left(h_b, D_t^m|_{m=1}^M\right) = 0$. Then Eq.(7) can be simplified as:

$$\begin{aligned} & L_t\left(H_t, D_t^m|_{m=1}^M\right) = \lambda Tr\left(H_t H_t^T\right) \\ & + \sum_{m=1}^M \alpha_m^2 Tr\left(H_{t\backslash b}\widetilde{K}_t^m H_{t\backslash b}^T - 2H_{t\backslash b}\left(Z_{t\backslash b}^m\right)^T + K_{t\backslash b}^m\right) \end{aligned} \tag{8}$$

After the dictionary is augmented, the new $\widetilde{K}_t^m$ and $Z_{t\backslash b}$ can be computed as:

$$\begin{aligned} \widetilde{K}_t^m &= \begin{bmatrix} \widetilde{K}_{t-1}^m & z_b^T \\ z_b & k_{bb} \end{bmatrix} \\ Z_{t\backslash b}^m &= \left[Z_{t-1\backslash b}^m, k\left(X_{t\backslash b}, x_b\right)\right] \end{aligned} \tag{9}$$

Lemma 1. *If $H_{t\backslash b} = \left[H_{t-1\backslash b}, \mathbf{0}\right]$, then the following equation is satisfied:*

$$L_t\left(H_{t\backslash b}, D_t^m|_{m=1}^M\right) = L_{t-1}\left(H_{t-1\backslash b}, D_{t-1}^m|_{m=1}^M\right) \tag{10}$$

Proof. From Eq.(9), we can obtain:

$$\begin{aligned} H_{t\backslash b}\widetilde{K}_t^m H_{t\backslash b}^T &= \left[H_{t-1\backslash b}, \mathbf{0}\right]\begin{bmatrix} \widetilde{K}_{t-1}^m & z_b^T \\ z_b & k_{bb} \end{bmatrix}\begin{bmatrix} H_{t-1\backslash b}^T \\ \mathbf{0} \end{bmatrix} \\ &= H_{t-1\backslash b}\widetilde{K}_{t-1}^m H_{t-1\backslash b}^T \end{aligned} \tag{11}$$

Similarly, we have:

$$H_{t\backslash b}\left(Z_{t\backslash b}^m\right)^T = H_{t-1\backslash b}\left(Z_{t-1\backslash b}^m\right)^T \tag{12}$$

Based on above two equations and Eq.(8), we can easily obtain that each element in L_t and L_{t-1} is equivalent, thus Eq.(10) is satisfied. □

Theorem 1. *The objective function $L_t\left(H_t, D_t^m|_{m=1}^M\right)$ will be decreased after we set $H_t = \left[H_{t\backslash b}^T, h_b^T\right]$, and $H_{t\backslash b} = \left[H_{t-1\backslash b}, \mathbf{0}\right]$ and $h_b = [\mathbf{0}, 1]$.*

Proof. According to Lemma 1 and Eq.(7), we can obtain that $L_t\left(H_t, D_t^m|_{m=1}^M\right) = L_{t-1}\left(H_{t-1\backslash b}, D_{t-1}^m|_{m=1}^M\right) + l_b\left(h_b, D_t^m|_{m=1}^M\right)$, and $l_b\left(h_b, D_t^m|_{m=1}^M\right) = 0$ can be satisfied by $h_b = [\mathbf{0}, 1]$. Before the dictionary augment, image b in the buffer always has a high $l_b\left(\bar{h}_b, D_{t-1}^m|_{m=1}^M\right)$ which describes the linear dependence, where $\bar{h}_b$ denote old codes of b. It is obviously that $l_b\left(\bar{h}_b, D_{t-1}^m|_{m=1}^M\right) > \rho > 0 = l_b\left(h_b, D_t^m|_{m=1}^M\right)$. Finally we have $L_t\left(H_t, D_t^m|_{m=1}^M\right) < L_{t-1}\left(H_{t-1}, D_{t-1}^m|_{m=1}^M\right)$. Therefore, the objective function can be decreased. □

Theorem 1 confirms that our objective function is consistently decreased by the buffer scheme. By specially considering the image with largest l_b and reduce it to 0 by using dictionary augment, the objective functions L_t can be largely decreased. If the buffer is full and we augment the codes, then we preserve half images with highest l to ensure that high l can be preserved for further process to decrease objective function. The frequency of augmenting depends on the buffer size, if buffer size is large, then the learning process is more efficient, but objective function can not be effectively decreased. Thus we should make a tradeoff between learning efficiency and effect by adjusting the buffer size.

Algorithm 1 Online learning process of DMVH at step t.

```
Input:
    $x_t^m|_{m=1}^M$,$D_{t-1}^m|_{m=1}^M$,$\widetilde{K}_{t-1}^m|_{m=1}^M$, $\alpha$
Output:
    $H_t$, $D_t^m|_{m=1}^M$, $\widetilde{K}_t^m|_{m=1}^M$, $\alpha$
Compute $h_t$ by Eq.(5);
Compute $l_t(h_t, D_{t-1}^m|_{m=1}^M)$ by Eq.(6)
if $l_t(h_t, D_{t-1}^m|_{m=1}^M) < \delta$ then
    $H_t = [H_{t-1}^T, sgn(h_t^T)]^T$;
    $\widetilde{K}_t^m = \widetilde{K}_{t-1}^m$ and $D_t^m = D_{t-1}^m$;
else if $l_t(h_t, D_{t-1}^m|_{m=1}^M) >= \delta$ then
    Add $t$ into the buffer, and use Algorithm 2 to optimize
    $H_t$, $\alpha$ and $\widetilde{K}_t^m$;
end if
```

Algorithm 2 Augment dictionary with buffer scheme.

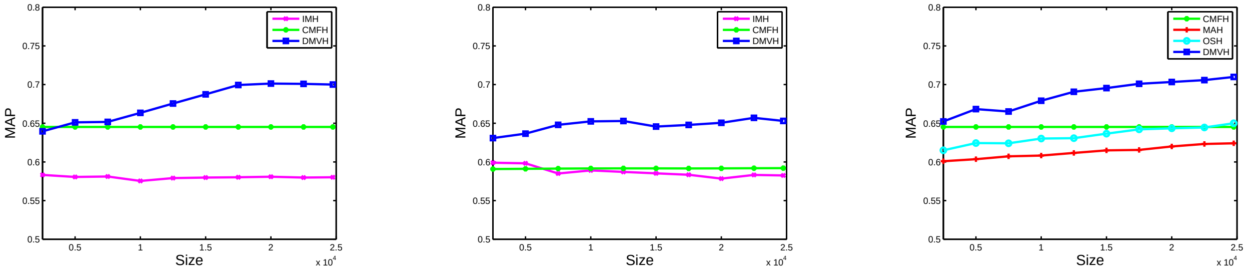
```
Input:
    Buffer, $H_t$, $D_{t-1}^m|_{m=1}^M$, $\widetilde{K}_{t-1}^m|_{m=1}^M$, $\alpha$
Output:
    $H_t$, $D_t^m|_{m=1}^M$, $\widetilde{K}_t^m|_{m=1}^M$, $\alpha$
Add image $t$ and corresponding $l_t(h_t, D_{t-1}^m|_{m=1}^M)$ into
buffer;
Buffer size $B = B + 1$;
if $B = B_{\max}$ then
    Select image $b$ in buffer with highest $l_b$;
    Remove $1/2$ of buffer data with lowest $l$;
    Augment dictionary $D_t^m = [(D_{t-1}^m)^T, (x_b^m)^T]^T$;
    Update $\widetilde{K}_t^m$ according to Eq.(9);
    Update $H_{t\backslash b} = \left[H_{t-1\backslash b}, \mathbf{0}\right]$;
    Update $h_b = [\mathbf{0}, 1]$;
    Update $\alpha$ according to Eq.(13)
else if $B < B_{\max}$ then
    $\widetilde{K}_t^m = \widetilde{K}_{t-1}^m$, $D_t^m = D_{t-1}^m$;
end if
```

At last, the view weights need to be updated to further reduce the objective function (7). With the Lagrange multiplier for Eq.(7), we can compute α_m using:

$$\alpha_m = \frac{1/\delta_m}{\sum_{i=1}^M 1/\delta_i} \tag{13}$$

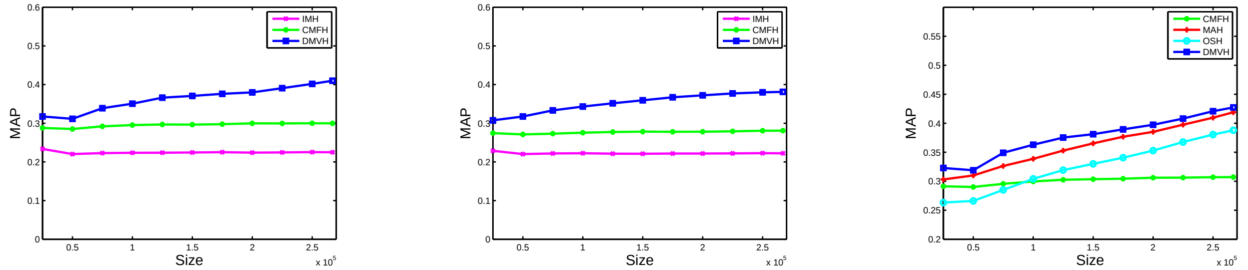
where $\delta_m = Tr\left(H_t\widetilde{K}_t^m H_t^T - 2H_t(Z_t^m)^T + K_{tt}^m\right)$.

The basic online learning process and dictionary augmentation process are shown in Algorithm 1 and 2 respectively. DMVH is efficient in learning hash codes. From Algorithm 1 we can find that the only the hash codes of new data is computed, and the old codes are not needed to update. Although Algorithm 2 updates the old hash codes of previous data, they



(a) Image query (b) Text query (c) Multi-view query

Figure 2: The MAP scores of MIR Flickr at different database sizes.



(a) Image query (b) Text query (c) Multi-view query

Figure 3: The MAP scores of NUS-WIDE at different database sizes.

are only augmented with all zero elements, which can be efficiently implemented. The time complexity of DMVH is independent of database size, it only relies on buffer size B and code length C.

Extension to visual and text features In real-world applications, users may not apply all the views as query but choose some of them. Besides the multi-view features, we specifically consider the query based on visual features or text feature. For the visual feature based query, its hash codes can be computed as: $h_v = \text{sgn}\left(\sum_{m=1}^{M-1} \alpha_m^2 z_v^m \left(\widetilde{K}_t\right)^{-1}\right)$, where view 1 to $M-1$ are the visual features and view M is text feature. If the query only contains text feature (e.g. several keywords), its hash codes can be computed as$h_{te} = \text{sgn}\left(z_{te}^M \left(\widetilde{K}_t\right)^{-1}\right)$.

4 Experiments

4.1 Datasets and Experimental Settings

Two large scale image test collections used for experimental study include:

- MIR Flickr [Huiskes and Lew, 2008] contains 25,000 images collected from Flickr. All images are annotated with 38 class labels which are used as the ground truth. In the retrieval process, images which share at least one same label are considered as relevant. 1% images are selected as queries, and the rest images are inserted to the database sequentially. 3 visual features [Guillaumin *et al.*, 2010] considered include 100-D Hue histogram, 1000-D SIFT BoVW, 4096-D RGB histogram and 512-D GIST. Each image is associated with text tags, thus 457-D BoW is used as text feature.
- NUS-WIDE [Chua *et al.*, 2009] includes 269,648 images collected from Flickr, each images is labeled by 81 concepts, which can be used for evaluation. Same as MIR Flickr, 1% images are used as queries and the rest images forms the streaming database. We also use 3 visual features, including 500-D SIFT BoVW, 64-D color histogram, 144-D color correlogram, 73-D edge direction histogram, 128-D wavelet texture and 225-D blockwise color. 1000-D BoW serves as text feature.

In the implementation of DMVH, we use Gaussian kernel for all visual features, and histogram intersection kernel for text feature. DMVH doesn't contain many parameters to configure. The regularization λ is set to 10^{-3}, it is used to avoid the matrix singularity and has little influence on the final results. The maximum buffer size is set to 1000 on MIR Flickr and 5000 on NUS-WIDE respectively, the threshold ρ is set to 0.5.

Since our method is purely based on unsupervised, we compare our method with several representative unsupervised hashing methods, including Online Sketching Hashing (OSH) [Leng *et al.*, 2015], Collective Matrix Factorization Hashing (CMFH) [Ding *et al.*, 2014], Multiview Alignment Hashing (MAH) [Liu *et al.*, 2015] and Inter-media Hashing (IMH) [Song *et al.*, 2013]. OSH cannot support any single-view or partial-view query such as text query and image query. CMFH can only cope with two views, thus we concatenate

Table 1: Learning time and code length at different sizes on NUS-WIDE.

Database size (10^4)	2.5	5	7.5	10	12.5	15	17.5	20	22.5	25	26.7
OSH time (s)	227	510	852	1269	1728	2248	2870	3610	4518	5844	7337
DMVH time (s)	135	287	469	684	932	1223	1558	1922	2317	2742	3099
DMVH code length	72	82	92	101	111	121	130	139	149	158	165

all visual features into a visual feature. MAH cannot support visual or text query, it only supports multi-view query. IMH supports visual and text query, but cannot support multi-view query.

Mean average precision (MAP) [Song *et al.*, 2013] is used to measure the effect of retrieval, and MAP scores are computed on the top 50 retrieved documents of each query. Moreover, we evaluate the learning time of all methods to measure the efficiency of retrieval. Learning time is computed as the total time of learning hash codes and functions, thus it accumulates the time of all learning processes from first step to current step. All the experiments are conducted on a computer with Intel Core(TM) i5 2.6GHz 2 processors and 12.0GB RAM.

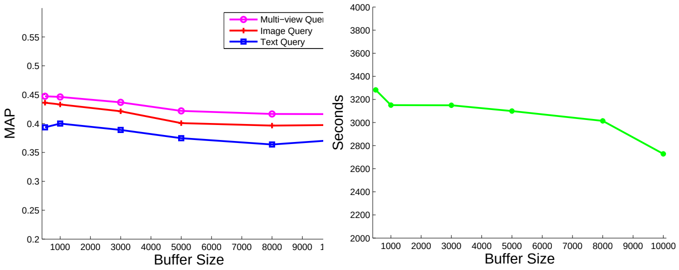
4.2 Results and Discussions

Performance Comparison: Experimental study considers three types of queries: visual query, text query and multi-view query. The visual query includes all the visual features, text query contains only one text feature, and multi-view query is the combination of visual and text features.

On MIR Flickr, images are added to database sequentially, thus the hashing performance is evaluated at different database size. We evaluate the MAP scores at $t = \{2.5, 5, 7.5, 10, 12.5, 15, 17.5, 20, 22.5, 25\} \times 10^3$. The code length of DVMH is changed with the increase of database size, therefore is not appropriate to evaluate the performance of other methods at various code lengths. According to previous results [Ding *et al.*, 2014], optimal code length we use is 64 bits. In order to ensure fair and robust comparison, DMVH doesn't increase the code length when it reaches 64 bits.

Figure 2 shows the MAP scores over MIR Flickr with different database sizes. In image query and text query, DMVH is compared with IMH and CMFH. In multi-view query, we compare DMVH with CMFH, MAH and OSH. From this figure we can find that DMVH obtains higher MAP scores than other methods. In visual and multi-view query, the MAP score of DMVH increases with the data size growth. This illustrates the effectiveness of our dynamic scheme where code length is adaptively augmented. With the augmentation of code length, the performance of DMVH can be consistently improved.

Similar to MIR Flickr, MAP is applied as evaluation metric on NUS-WIDE with different test collection sizes, the optimal code length of compared methods is set to 128 bits. The upper bound of code length in DMVH is set to be 128 bits. Figure 3 shows the MAP scores over different database sizes. We observe similar result as with MIR Flickr, DMVH outperforms other multi-view hashing methods. On all types of queries, the MAP socres of DMVH are improved with the increase of data size. The increase of MAP scores on NUS-WIDE are more significant than MIR Flickr, the reason is that NUS-WIDE contains much more images, thus the increase of code length are more benefit to NUS-WIDE. Note that on NUS-WIDE, we use all 81 labels for evaluation, other works such as [Ding *et al.*, 2014] use a reduced version which only contains 10 labels. Therefore the overall scores reported in this paper are relatively lower.



(a) MAP Scores

(b) Training time

Figure 4: Effects of buffer size on NUS-WIDE.

Efficiency Analysis: We also compare the learning time of DMVH to online hashing method OSH. Table 1 reports the learning time of two methods, and the code length of DMVH at different sizes. We can find that DMVH consumes less learning time than OSH. DMVH uses the buffer scheme to avoid the frequent updating process. It is more efficient than OSH while guarantees the hashing performance. In addition, we can find that the code length of DMVH is dynamically changed with the increase of data size.

Effects of Buffer: Table 4 reports the comparison of MAP and training time with buffer size {500,1000,3000, 5000, 7000, 10000}. All the results are evaluated at the final step of NUS-WIDE, where all 269648 images are included in database. We can find that the results are relatively steady at different buffer size, which demonstrates the robustness of DMVH. Moreover, both MAP scores and training time decrease with the buffer size, thus we can choose a proper size by considering the tradeoff between MAP scores and training time.

5 Conclusions

This paper presents Dynamic Multi-view Hashing (DMVH) for online retrieval of streaming image. Distinguished from existing approaches, DMVH can adaptively augment code length to preserve more discriminative information of new image. It constructs hash codes by a dynamic dictionary, when new image cannot be effectively represented using existing hash codes. Moreover, DMVH can augment the dictionary to give better representation of this data. To avoid the frequent updating of the dictionary, we also design a buffer scheme, where the only data significantly different to existing elements is considered for updating. Experimental results on MIR Flickr and NUS-WIDE demonstrate the superior efficiency and effectiveness of DMVH over the state of the art online hashing and multi-view hashing methods.

References

[Caicedo *et al.*, 2012] Juan C Caicedo, Jaafar BenAbdallah, Fabio A González, and Olfa Nasraoui. Multimodal representation, indexing, automated annotation and retrieval of image collections via non-negative matrix factorization. *Neurocomputing*, 76(1):50–60, 2012.

[Cakir and Sclaroff, 2015a] Fatih Cakir and Stan Sclaroff. Adaptive hashing for fast similarity search. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1044–1052, 2015.

[Cakir and Sclaroff, 2015b] Fatih Cakir and Stan Sclaroff. Online supervised hashing. In *Image Processing (ICIP), 2015 IEEE International Conference on*, pages 2606–2610. IEEE, 2015.

[Chua *et al.*, 2009] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval*, page 48. ACM, 2009.

[Ding *et al.*, 2014] Guiguang Ding, Yuchen Guo, and Jile Zhou. Collective matrix factorization hashing for multimodal data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2075–2082, 2014.

[Guillaumin *et al.*, 2010] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Multimodal semi-supervised learning for image classification. In *CVPR 2010-23rd IEEE Conference on Computer Vision & Pattern Recognition*, pages 902–909. IEEE Computer Society, 2010.

[Huang *et al.*, 2013] Long-Kai Huang, Qiang Yang, and Wei-Shi Zheng. Online hashing. In *IJCAI*, 2013.

[Huiskes and Lew, 2008] Mark J Huiskes and Michael S Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43. ACM, 2008.

[Kim and Choi, 2013] Saehoon Kim and Seungjin Choi. Multi-view anchor graph hashing. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 3123–3127. IEEE, 2013.

[Kumar and Udupa, 2011] Shaishav Kumar and Raghavendra Udupa. Learning hash functions for cross-view similarity search. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1360, 2011.

[Leng *et al.*, 2015] Cong Leng, Jiaxiang Wu, Jian Cheng, Xiao Bai, and Hanqing Lu. Online sketching hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2503–2511, 2015.

[Liu *et al.*, 2015] Li Liu, Mengyang Yu, and Ling Shao. Multiview alignment hashing for efficient image search. *Image Processing, IEEE Transactions on*, 24(3):956–966, 2015.

[Rasiwasia *et al.*, 2010] Nikhil Rasiwasia, Jose Costa Pereira, Emanuele Coviello, Gabriel Doyle, Gert RG Lanckriet, Roger Levy, and Nuno Vasconcelos. A new approach to cross-modal multimedia retrieval. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 251–260. ACM, 2010.

[Song *et al.*, 2013] Jingkuan Song, Yang Yang, Yi Yang, Zi Huang, and Heng Tao Shen. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pages 785–796. ACM, 2013.

[Wu *et al.*, 2014] Fei Wu, Zhou Yu, Yi Yang, Siliang Tang, Yin Zhang, and Yueting Zhuang. Sparse multi-modal hashing. *Multimedia, IEEE Transactions on*, 16(2):427–439, 2014.

[Wu *et al.*, 2015] Botong Wu, Qiang Yang, Wei-Shi Zheng, Yizhou Wang, and Jingdong Wang. Quantized correlation hashing for fast cross-modal search. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, 2015.

[Yu *et al.*, 2014] Zhou Yu, Fei Wu, Yi Yang, Qi Tian, Jiebo Luo, and Yueting Zhuang. Discriminative coupled dictionary hashing for fast cross-media retrieval. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 395–404. ACM, 2014.

[Zhang and Li, 2014] Dongqing Zhang and Wu-Jun Li. Large-scale supervised multimodal hashing with semantic correlation maximization. In *AAAI*, pages 2177–2183, 2014.

[Zhang *et al.*,] Yin Zhang, Weiming Lu, Yang Liu, and Fei Wu. Kernelized sparse hashing for scalable image retrieval. *Neurocomputing*, 172.

[Zhang *et al.*, 2011] Dan Zhang, Fei Wang, and Luo Si. Composite hashing with multiple information sources. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 225–234. ACM, 2011.

[Zhen and Yeung, 2012] Yi Zhen and Dit-Yan Yeung. A probabilistic model for multimodal hash function learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 940–948. ACM, 2012.

[Zhou *et al.*, 2014] Jile Zhou, Guiguang Ding, and Yuchen Guo. Latent semantic sparse hashing for cross-modal similarity search. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 415–424. ACM, 2014.