Dynamic Multi-View Hashing for Online Image Retrieval

Liang Xie[1], **Jialie Shen**[2*], **Jungong Han**[3], **Lei Zhu**[4], **Ling Shao**[5]
[1]Wuhan University of Technology, China
[2]Northumbria University, United Kingdom
[3]Lancaster University, United Kingdom
[4]The University of Queensland, Australia
[5]University of East Anglia, United Kingdom
whutxl@hotmail.com, {jialie, jungonghan77, leizhu0608}@gmail.com, ling.shao@ieee.org

Abstract

Advanced hashing technique is essential in large scale online image retrieval and organization, where image contents are frequently changed. While traditional multi-view hashing method has achieve promising effectiveness, its batch-based learning based scheme largely leads to very expensive updating cost. Meanwhile, existing online hashing scheme generally focuses on single-view data. Good effectiveness can not be expected when searching over real online images, which typically have multiple views. Further, both types of hashing methods only can generate hash codes with fixed length. Thus they have limited capability on comprehensive characterization of streaming image 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 in the online learning process. In order to avoid the frequent updating of code length and view weights, an intelligent buffering scheme is 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, techniques to facilitate fast and effective large scale content access are getting more and more attentions. In recent years, hashing has emerged as a promising technology to support large-scale image retrieval [Mu *et al.*, 2010; Zhen and Yeung, 2012a; Guo *et al.*, 2017b; Wu *et al.*, 2017; Zhu *et al.*, 2017]. Except various visual features, the Web images are associated with various text information such as tags or short comments. In order to improve Web image search performance, various multi-view hashing methods have been recently proposed [Zhang *et al.*, 2011; Wu *et al.*, 2014; 2015; Xie *et al.*, 2016; Liu *et al.*, 2017; Lin *et al.*, 2017b; 2017a; Guo *et al.*, 2017a]. Multi-view hashing integrates the advantages of hashing technology and multi-view learning, it enjoys better performance than single-view hashing, and can support the queries from different views such as visual features or text features.

Web image database is highly dynamic and frequently updated. Traditional multi-view hashing methods are developed based on batch-based learning, which is very expensive in terms of updating cost. Multi-view hashing methods may accumulate all data to retrain the hash codes and functions when new images arrive, however their time complexity will be extremely high and unaffordable. Recently, several online hashing methods [Leng *et al.*, 2015; Cakir and Sclaroff, 2015b] are proposed to support effective search over dynamic image databases. However, their performance is far beyond satisfactory. When applying existing multi-view hashing and online hashing methods for web image retrieval, they generally suffer from several issues. Existing online hashing methods aim at generating the binary codes with fixed length, which is not suited to the dynamic image data. In real world, the contents of new image data could be semantically different from the old one. When new image data is inserted into the database, longer codes should be required to preserve more discriminative information to support search over updated image database. According to previous results [Zhou *et al.*, 2014], longer hash code can lead to more comprehensive image representation and better search performance. For example, Wikipedia dataset [Rasiwasia *et al.*, 2010] consisting of 2866 images only needs 32 bits codes to achieve the best performance. However, for NUS-WIDE having 269648 images, at least 128 bits are required to achieve 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, 2012b]. Therefore, general hashing methods usually choose an appropriate code length for a particular database, which is not suitable to dynamic database, whose size is continuously changed. To solve this problem, the code length for dynamic image

*Corrsponding Author

database needs to be intelligently augmented to guarantee hashing performance. Another problem is how to measure importance of different views in the online learning of hash codes. Although each view will contribute to the final hash codes, they can not be equally treated. For example, text feature usually contains more semantic information than visual feature [Caicedo *et al.*, 2012], thus it should be assigned with the higher weight. Several multi-view hashing methods take the view weights into account and thus they achieve significant performance improvement. However, 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: Dynamic Multi-view Hashing (DMVH) to support the effective and efficient dynamic online image retrieval. Towards this goal, DMVH uses a dynamic hashing scheme, where length of hashing codes can be adaptively augmented. In DMVH, a dictionary is used to construct the hash codes of images. When a new data cannot be represented by current dictionary, it will be added to augment the dictionary. Moreover, DMVH applies multi-view features for hashing, and proper weight is assigned to each view. In the online learning process, to avoid the frequent updating of dictionary and modality weights, a buffer is used to preserve significant images for purpose of subsequential updating. The core technical contributions of our work can be summarized as follows:

- 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.
- DMVH can effectively estimate proper weights for different views to reflect their different importance in hashing. Previous online hashing methods generally don't have weighting scheme. An intelligent buffer scheme is designed for the online learning process of DMVH, which enables highly efficient learning process of hash codes and view weights.
- Experimental results on real-world image datasets show the superiority of DMVH in terms of both search efficiency and accuracy.

The remainder of this paper is organized as follows: Section 2 introduces related work; Section 3 comprehensively introduces the proposed multi-view hashing (DMVH) method; Section 4 introduces the experimental configurations and reports the experimental results and main findings. Finally, Section 5 concludes the paper with future work.

2 Related Work

Multi-view Hashing - In recent years, much attention has been paid for multi-view hashing. Most multi-view hashing methods can be divided to two categories according to the query types supported. Several multi-view hashing methods combine all features to construct database hash codes, but they require that query has the same features as database

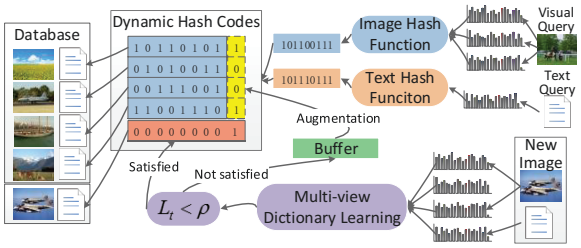


Figure 1: The overall illustration of DMVH.

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] uses graphs of multi-views features to learn the hash codes, and each view is assigned with a weight for combination. Since graph learning is inefficient, anchor graph is adopted to accelerate the learning process. Multi-view Anchor Graph Hashing (MVAGH) [Kim and Choi, 2013] exploits anchors to construct graphs for 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 methods also can support queries based on different views. This type of multi-view hashing is also called as cross-view hashing. Generally, cross-view hashing methods aim at modeling the correlation of different views. Cross-view Hashing (CVH) [Kumar and Udupa, 2011] optimizes the hamming distance of different views, it can be regarded as an extended version of Canonical Correlation Analysis (CCA). 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 correlates them in a common hash space.

Online Hashing is more practical than traditional batch-based learning methods, in that many new images in real-world are generated. In recent three years, a few online hashing methods are proposed. Online Kernel Hashing (OKH) [Huang *et al.*, 2013] and Adaptive Hashing [Cakir and Sclaroff, 2015a] are supervised methods, they use similar image pairs as training data. Although they can effectively preserve the correlation of images in hash codes, they require labeled training examples which are difficult to collect. Online Sketching Hashing (OSH) [Leng *et al.*, 2015] is unsupervised online hashing method, thus it is not relied on labeled data. The main disadvantage of existing online hashing methods is that they are not designed for multi-view features, so they cannot perform well for the retrieval of multi-view images.

3 Dynamic Multi-View Hashing

DMVH consists of two main modules: online learning process and search process. Figure 1 illustrates the overall structure of DMVH. 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 consists of many 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 by 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 $D_t^m = [(x_{a_1}^m)^T, \ldots, (x_{a_C}^m)^T]^T$. 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, $\alpha = [\alpha_1, \ldots, \alpha_M]$ 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 to database images and cannot be represented by current dictionary, we have to frequently update the dictionary, which is 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 added 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:

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

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:

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

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

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

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

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

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

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

Similarly, we have:

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

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.

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

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

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

Algorithm 1 Online learning process of DMVH at step t.

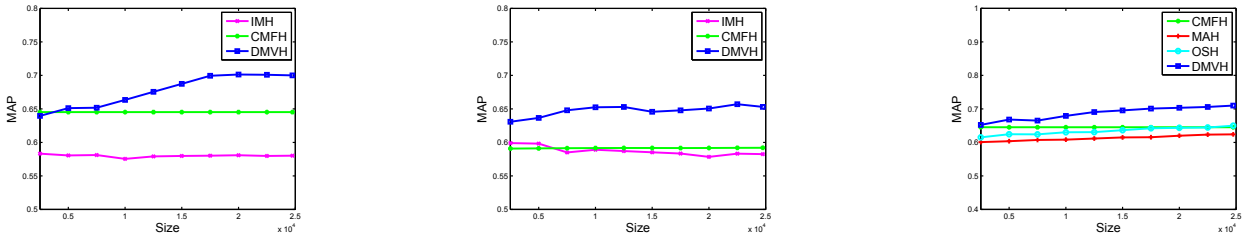
Input:
$x_t^m|_{m=1}^M$,$D_{t-1}^m|_{m=1}^M$,$\widetilde{K}_{t-1}^m|_{m=1}^M$, α
Output:
H_t, $D_t^m|_{m=1}^M$, $\widetilde{K}_t^m|_{m=1}^M$, α

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, α and $\widetilde{K}_t^m$;
end if

Algorithm 2 Augment dictionary with buffer scheme.

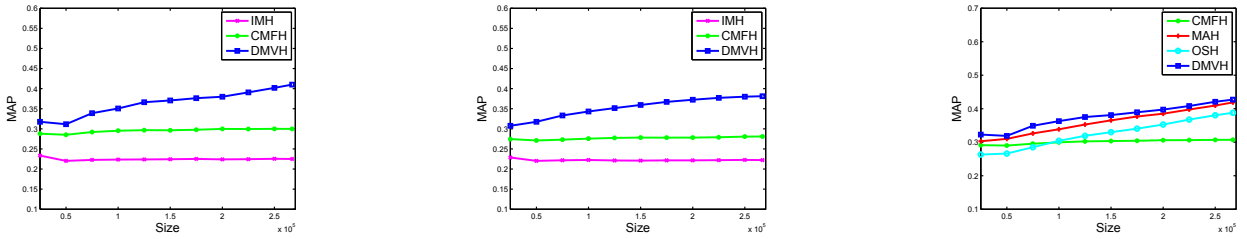
Input:
Buffer, H_t, $D_{t-1}^m|_{m=1}^M$, $\widetilde{K}_{t-1}^m|_{m=1}^M$, α
Output:
H_t, $D_t^m|_{m=1}^M$, $\widetilde{K}_t^m|_{m=1}^M$, α

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 α 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



(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.

data, they 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 use all the views as query and be likely to choose parts of them. Besides the multi-view features, we specifically consider the case of visual features and only text feature. For the query based on visual features, its hash codes can be computed as:$h_v = \operatorname{sgn}\left(\sum_{m=1}^{M-1} \alpha_m^2 z_v^m \left(\widetilde{K}_t\right)^{-1}\right)$. 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), we can compute its hash codes as$h_{te} = \operatorname{sgn}\left(z_{te}^M \left(\widetilde{K}_t\right)^{-1}\right)$.

4 Experiments

4.1 Datasets

We use two multi-view image datasets: MIR Flickr [Huiskes and Lew, 2008] and NUS-WIDE [Chua *et al.*, 2009] to compare the hashing performance in the online scenario.

- MIR Flickr contains 25000 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. We select 1% images as queries, and the rest images are added to the database sequentially. We use 3 visual features [Guillaumin *et al.*, 2010], including 100-D Hue histogram, 1000-D SIFT BoVW, 4096-D RGB histogram and 512-D GIST. Each image is associated with text tags, thus we use 457-D BoW as text feature.
- NUS-WIDE contains 269648 images collected from Flickr, each images is labeled by 81 concepts which can be used for evaluation. We select 1% images as queries and the rest images forms the streaming database. We 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 block-wise color. Each image is associated with text tags, thus we also use 1000-D BoW as text feature.

4.2 Experimental Settings

In the implementation of DMVH, we use Gaussian kernel for all visual features, and histogram intersection kernel for text feature. DMVH does not contain many parameters to set. The regularization λ is set to 10^{-3}, it is used to avoid the matrix singularity and has little influence on the 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 methods does not use any supervised information, 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 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

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

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.

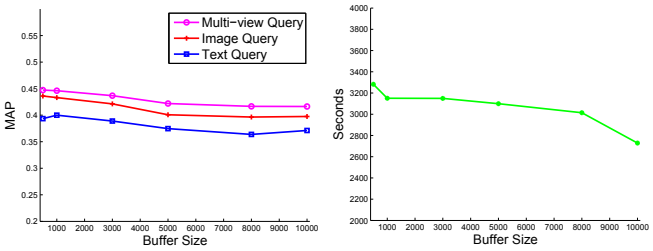
4.3 Experimental Results

In the experiments, we consider three types of queries: visual query, text query and multi-view query. The visual query contains all visual features, text query contains only one text feature, and multi-view query consists of all features.

On MIR Flickr, images are added to database sequentially, thus we have to evaluate the hashing performance at different database size. More specifically, 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], we choose the best code length as 64 bits. In order to make a fair comparison, DMVH will not increase the code length when it reaches 64 bits.

Figure 2 shows the MAP scores of MIR Flickr at different database sizes. In image query and text query, we compare DMVH 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 image and multi-view query, the MAP scores DMVH are increased with the data size. This phenomenon 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, on NUS-WIDE we evaluate the MAP scores at different sizes, the best code length of compared methods is set to 128 bits. The code length of DMVH will not increase when it reaches 128 bits. Figure 3 shows the MAP scores at different database sizes. Similar to the results of 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.

4.4 Efficiency Analysis

we also compare the learning time of DMVH to online hashing method OSH. Table 1 shows 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.

4.5 Effects of Buffer

At last we evaluate the effects of buffer size. Table 4 shows 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

In this paper we propose Dynamic Multi-view Hashing (DMVH) for online retrieval of streaming image data. DMVH can adaptively augment code length to preserve more information of new image. It constructs hash codes by a dynamic dictionary, when new data cannot be effectively represented by current hash codes, DMVH can augment the dictionary to better represent this data. To avoid the frequent updating of dictionary. A buffer scheme is designed and only data which are significantly different to current dictionary elements are considered for updating. Experimental results on MIR Flickr and NUS-WIDE demonstrate the efficiency and effectiveness of DMVH compared to representative 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 *CVPR*, pages 1044–1052, 2015.

[Cakir and Sclaroff, 2015b] Fatih Cakir and Stan Sclaroff. Online supervised hashing. In *ICIP*, pages 2606–2610, 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 *ACM CIVR*, page 48, 2009.

[Ding *et al.*, 2014] Guiguang Ding, Yuchen Guo, and Jile Zhou. Collective matrix factorization hashing for multimodal data. In *CVPR*, pages 2075–2082, 2014.

[Guillaumin *et al.*, 2010] Matthieu Guillaumin, Jakob Verbeek, and Cordelia Schmid. Multimodal semi-supervised learning for image classification. In *CVPR*, pages 902–909, 2010.

[Guo *et al.*, 2017a] Y. Guo, G. Ding, J. Han, and Y. Gao. Zero-shot learning with transferred samples. *IEEE Trans. Image Processing*, 26(17):3277–3290, 2017.

[Guo *et al.*, 2017b] Yuchen Guo, Guiguang Ding, Li Liu, Jungong Han, and Ling Shao. Learning to hash with optimized anchor embedding for scalable retrieval. *IEEE Trans. Image Processing*, 26(3):1344–1354, 2017.

[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 *ACM ICMR*, pages 39–43, 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, 2013.

[Kumar and Udupa, 2011] Shaishav Kumar and Raghavendra Udupa. Learning hash functions for cross-view similarity search. In *IJCAI*, page 1360, 2011.

[Leng *et al.*, 2015] Cong Leng, Jiaxiang Wu, Jian Cheng, Xiao Bai, and Hanqing Lu. Online sketching hashing. In *CVPR*, pages 2503–2511, 2015.

[Lin *et al.*, 2017a] Z. Lin, G. Ding, J. Han, and L. Shao. End-to-end feature-aware label space encoding for multilabel classification with many classes. *IEEE Trans. Neural Networks and Learning Systems*, 2017.

[Lin *et al.*, 2017b] Z. Lin, G. Ding, J. Han, and J. Wang. Cross-view retrieval via probability-based semantics-preserving hashing. *IEEE Trans. Cybernetics*, 2017.

[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.

[Liu *et al.*, 2017] Li Liu, Zijia Lin, Ling Shao, Fumin Shen, Guiguang Ding, and Jungong Han. Sequential discrete hashing for scalable cross-modality similarity retrieval. *IEEE Trans. Image Processing*, 26(1):107–118, 2017.

[Mu *et al.*, 2010] Yadong Mu, Jialie Shen, and Shuicheng Yan. Weakly-supervised hashing in kernel space. In *CVPR*, pages 3344–3351, 2010.

[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 *ACM MM*, pages 251–260, 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 *ACM SIGMOD*, pages 785–796, 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 *IJCAI*, 2015.

[Wu *et al.*, 2017] Gengshen Wu, Li Liu, Yuchen Guo, Guiguang Ding, Jungong Han, Jialie Shen, and Ling Shao. Unsupervised deep video hashing with balanced rotation. In *IJCAI*, 2017.

[Xie *et al.*, 2016] Liang Xie, Jialie Shen, and Lei Zhu. Online cross-modal hashing for web image retrieval. In *AAAI*, pages 294–300, 2016.

[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 *ACM SIGIR*, pages 395–404, 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 *ACM SIGIR*, pages 225–234, 2011.

[Zhen and Yeung, 2012a] Yi Zhen and Dit-Yan Yeung. Co-regularized hashing for multimodal data. In *NIPS 25*, pages 1376–1384. 2012.

[Zhen and Yeung, 2012b] Yi Zhen and Dit-Yan Yeung. A probabilistic model for multimodal hash function learning. In *ACM SIGKDD*, pages 940–948, 2012.

[Zhou *et al.*, 2014] Jile Zhou, Guiguang Ding, and Yuchen Guo. Latent semantic sparse hashing for cross-modal similarity search. In *ACM SIGIR*, pages 415–424, 2014.

[Zhu *et al.*, 2017] Lei Zhu, Jialie Shen, Liang Xie, and Zhiyong Cheng. Unsupervised visual hashing with semantic assistant for content-based image retrieval. *IEEE Trans. Knowl. Data Eng.*, 29(2):472–486, 2017.