Dynamic Multi-View Hashing for Online Image Retrieval

Liang Xie¹, Jialie Shen²*, Jungong Han³, Lei Zhu⁴, Ling Shao⁵
¹Wuhan University of Technology, China
²Northumbria University, United Kingdom
³Lancaster University, United Kingdom
⁴The University of Queensland, Australia
⁵University of East Anglia, United Kingdom
whutexl@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 achieved promising effectiveness, its batch-based learning 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 DMVH 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 attention. 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
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 predefined 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 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 hash 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 in-media and inter-media consistency. Semantic Correlation Maximization (SCM) [Zhang and Li, 2014] maximizes the semantic correlation between image and text features, and 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 $X^m_{t-1}|_{m=1}^M$, where $M$ is the number of views. $X^m_{t-1} \in \mathbb{R}^{(t-1) \times d_m}$, 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 $x^m_t$ is added into the database, and $X^m_t = [(X^m_{t-1})^T, (x^m_t)^T]^T$. 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 acquire more discriminative information. Thus, at time $t$, the dictionary consists of a subset of the database samples with multi-view features. For each view, its corresponding dictionary is denoted as $D^m_t = [(x^m_{a_1}), \ldots, (x^m_{a_C})]^T$. Assuming each sample can be constructed from its hash codes and dictionary, we have the following formulation:

$$\min L_t(H_t, D^m_t|_{m=1}^M) = \sum_{i=1}^M \alpha^2_m \|H_t\phi(D^m_i) - \phi(x^m_t)\|^2_F + \lambda \|H_t\|^2_F$$

s.t. $\sum_{m=1}^M \alpha_m = 1$ (1)

where $\phi(\cdot)$ is a kernel projection function, $\lambda$ 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:

$$\min L_t(H_t, D^m_t|_{m=1}^M) = L_{t-1}(H_{t-1}, D^m_t|_{m=1}^M) + l_t(H_t, D^m_t|_{m=1}^M)$$

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

$$\min l_t(h_t, D^m_t|_{m=1}^M) = \sum_{m=1}^M \alpha^2_m \|h_t\phi(D^m_i) - \phi(x^m_t)\|^2_F + \lambda \|h_t\|^2_F$$

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

$$\sum_{m=1}^M \alpha^2_m \left(h_tK^m_t - 2h_t(z^m_t)^T + k^m_t\right) + \lambda h_t^T h_t$$ (4)

where $K^m_t \in \mathbb{R}^{C \times C}$ is the kernel matrix of the dictionary $D^m_t$, and $K^m_t(i,j) = \phi(x^m_{a_i})\phi(x^m_{a_j})^T$, $z^m_t = \phi(x^m_t)\phi(x^m_t)^T$, $a_i$ is the index of $i_{th}$ dictionary element. $k^m_t = \phi(x^m_t)\phi(x^m_t)^T$.

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

$$h_t = z_t(K_t + \lambda I)^{-1}$$ (5)

where $K_t = \sum_{m=1}^M \alpha^2_m K^m_t$, $z_t = \sum_{m=1}^M \alpha^2_m z^m_t$.

For each modality $m$, if vector $\phi(x^m_t)$ is linearly independent on vectors of dictionary $\phi(D^m_t)$. Then $h_t$ computed by Eq.(5) cannot satisfy Eq.(3), and $x^m_t$ 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:

$$l_t(h_t, D^m_t|_{m=1}^M) = \sum_{m=1}^M \alpha^2_m (k^m_t - z^m_t h_t) + \lambda h_t^T h_t$$ (6)

We use a threshold $\rho$ to determine the linear dependence. If $l_t(h_t, D^m_t|_{m=1}^M) < \rho$, then the approximate linear dependence described by Eq.(3) can be satisfied. And we do not need to change dictionary. If $l_t(h_t, D^m_t|_{m=1}^M) \geq \rho$, 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 $l_t(h_t, D^m_t|_{m=1}^M) \geq \rho$, 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 $\alpha$. The dictionary is updated with $D^m_t = [(D^m_{t-1})^T, (x^m_t)^T]^T$, and the hash codes constructed from dictionary have to be augmented accordingly. We should optimize the following overall objective function:

$$\min L_t(H_t, D^m_t|_{m=1}^M) = L_{t\setminus b}(H_{t\setminus b}, D^m_t|_{m=1}^M) + l_b(h_b, D^m_t|_{m=1}^M)$$

s.t. $\sum_{m=1}^M \alpha_m = 1$ (7)
Where $H_{t,b}$ 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 $h_b = \{0, 1\}$, and $l_b (h_b, D_{t-1}^{|M|_{m=1}}) = 0$. Then Eq.(7) can be simplified as:

$$L_t (H_t, D_t^{|M|_{m=1}}) = \alpha Tr (H_t H_t^T) + \sum_{m=1}^M \alpha^2 m Tr (H_{t,b} K_{t,b} H_{t,b}^T - 2H_{t,b}(Z_{t,b}^m)^T + K_{t,b})$$

(8)

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

$$\tilde{K}_t^m = \begin{bmatrix} \tilde{K}_t^m_{1-1} & z_t^m T_b \\ z_b^m & k_{bb} \end{bmatrix}$$

$$Z_{t,b}^m = \begin{bmatrix} Z_{t-1,b}^m, k \left( X_{t,b}, x_b \right) \end{bmatrix}$$

(9)

Lemma 1. If $H_{t,b} = [H_{t-1,b}, 0]$, then the following equation is satisfied:

$$L_t (H_{t,b}, D_t^{|M|_{m=1}}) = L_{t-1} (H_{t-1,b}, D_{t-1}^{|M|_{m=1}})$$

(10)

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

$$H_{t,b} \tilde{K}_t^m H_{t,b}^T = [H_{t-1,b}, 0] \begin{bmatrix} \tilde{K}_t^m_{1-1} & z_t^m T_b \\ z_b^m & k_{bb} \end{bmatrix} \begin{bmatrix} H_{t-1,b}^T \\ 0 \end{bmatrix}$$

= $H_{t-1,b} \tilde{K}_t^m_{1-1} H_{t-1,b}^T$

(11)

Similarly, we have:

$$H_{t,b}^T Z_{t,b}^m T = H_{t-1,b}^T Z_{t-1,b}^m T$$

(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 $L_t (H_t, D_t^{|M|_{m=1}})$ will be decreased after we set $H_t = [H_{t-1,b}, h_b^T]$, and $H_{t,b} = [H_{t-1,b}, 0]$ and $h_b = \{0, 1\}$.

Proof. According to Lemma 1 and Eq.(7), we can obtain that $L_t (H_t, D_t^{|M|_{m=1}}) = L_{t-1} (H_{t-1,b}, D_{t-1}^{|M|_{m=1}}) + l_b (h_b, D_{t}^{|M|_{m=1}}), and l_b (h_b, D_{t}^{|M|_{m=1}}) = 0$ can be satisfied by $h_b = \{0, 1\}$. Before the dictionary augment, image $b$ in the buffer always has a high $l_b (h_b, D_{t}^{|M|_{m=1}})$ which describes the linear dependence, where $h_b$ denote old codes of $b$. It is obviously that $l_b (h_b, D_{t}^{|M|_{m=1}}) > \alpha > 0 = l_b (h_b, D_{t-1}^{|M|_{m=1}})$. Finally we have $L_t (H_t, D_t^{|M|_{m=1}}) < L_{t-1} (H_{t-1}, D_{t-1}^{|M|_{m=1}})$. 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 $\alpha_m$ by:

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

(13)

where $\delta_m = Tr \left( H_t \tilde{K}_t^m H_t^T - 2H_t Z_t^m T + K_t^m \right)$. 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 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:

$$h_v = \text{sgn} \left( \sum_{m=1}^{M-1} \alpha_m z^m \left( \tilde{K}_t \right)^{-1} \right).$$

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:

$$h_{te} = \text{sgn} \left( z^M \left( \tilde{K}_t \right)^{-1} \right).$$

**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 image 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 $\lambda$ 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 $\rho$ 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 can 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
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 \( t = \{2.5, 5, 7.5, 10, 12.5, 15, 17.5, 20, 22.5, 25\} \times 10^4 \). The code length of DMVH 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 buffer sizes for the 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 scores 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.

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 Flicker and NUS-WIDE demonstrate the efficiency and effectiveness of DMVH compared to representative online hashing and multi-view hashing methods.

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

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

(a) MAP Scores

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


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


