Unsupervised Deep Video Hashing with Balanced Rotation

IJCAI Anonymous Submission 2367

Abstract

Recently, hashing video contents for fast retrieval has received increasing attention due to the enormous growth of online videos. As the extension of image hashing techniques, traditional video hashing methods mainly focus on seeking the appropriate video features but pay little attention to how the video-specific features can be leveraged to achieve optimal binarization. In this paper, an end-to-end hashing framework, namely Unsupervised Deep Video Hashing (UDVH), is proposed, where feature extraction, balanced code learning and hash function learning are integrated and optimized in a self-taught manner. Particularly, distinguished from previous work, our framework enjoys two novelties: 1) an unsupervised hashing method that integrates the feature clustering and feature binarization, enabling the neighborhood structure to be preserved in the binary space; 2) a smart rotation applied to the video-specific features that are widely spread in the low-dimensional space such that the variance of dimensions can be balanced, thus generating more effective hash codes. Extensive experiments have been performed on two real-world datasets and the results demonstrate its superiority, compared to the state-of-the-art video hashing methods. To bootstrap further developments, the source code will be made publically available.

1 Introduction

Hashing, also known as binary coding, has been an effective method for fast nearest neighbor search due to the low storage requirement and the high calculation efficiency of the XOR operation in the Hamming Space [Gong and Lazebnik, 2011; Xu *et al.*, 2013]. As a result, recent decade has witnessed its broad range of multimedia computing applications, such as Content-based Video Retrieval (CBVR).

Early-stage video hashing algorithms are prone to extracting frame-level features from the video so that the available image hashing techniques can be plugged in directly. Two representatives are Multiple Feature Hashing [Song *et al.*, 2011] and Submodular Video Hashing [Cao *et al.*, 2012], where the former encodes the extracted key frames while the latter characterizes the video by averaging features of individual frames. Later on, temporal features over successive frames are considered, including motion trajectory [Wang and Schmid, 2013] and temporal consistency [Ye *et al.*, 2013]. This sort of features is proved to boost the performance even though temporal information might be partially lost during the frame pooling and dimensionality reduction.

Inspired by the great success of deep learning in image recognition, several recent systems have incorporated the hashing functions into the deep learning architecture. For example, Liong et al. [Liong *et al.*, 2016] train a deep network to exploit the discriminative and temporal information of video, assuming the video pairwise information is available. Although this supervised deep network is capable of generating similar binary codes for videos belonging to the same category, labeling the training data is rather expensive, which hinders it from being used in the real-world applications. Alternatively, Zhang et al. [Zhang *et al.*, 2016] propose a deep encoder-decoder framework, where a two-layer Long Short Term Memory (LSTM) unit followed by a binarization layer is able to directly encode the video features into compact binary codes. Minimizing feature reconstruction distortion allows feature extraction and binarization to be jointly optimized. Apart from the fact that such an encoder-decoder framework is quite complicated due to the involvement of de-binarization and de-LSTM, the objective function based on minimizing reconstruction error does not seem to preserve the neighborhood structure of data, which is important for a similarity search task.

It can be observed that most existing video hashing techniques focus on seeking appropriate video features and wrapping them up as image features so that the binarization developed for image hashing can be adopted. In other words, there is no binarization specifically designed for the video features. Compared to the image features, video features normally acquire the temporal information of a video sequence, such as the video dynamics extracted by LSTM, which are likely to be more scattered (less correlated) in terms of distribution [Karpathy *et al.*, 2016]. When projecting these scattered features into the low-dimensional compact space (e.g., using PCA), which is a necessary step of binarization, the data variances on projected dimensions tend to be large. This imbalanced projection will degrade the performance of the generated hash codes because each dimension will be treated

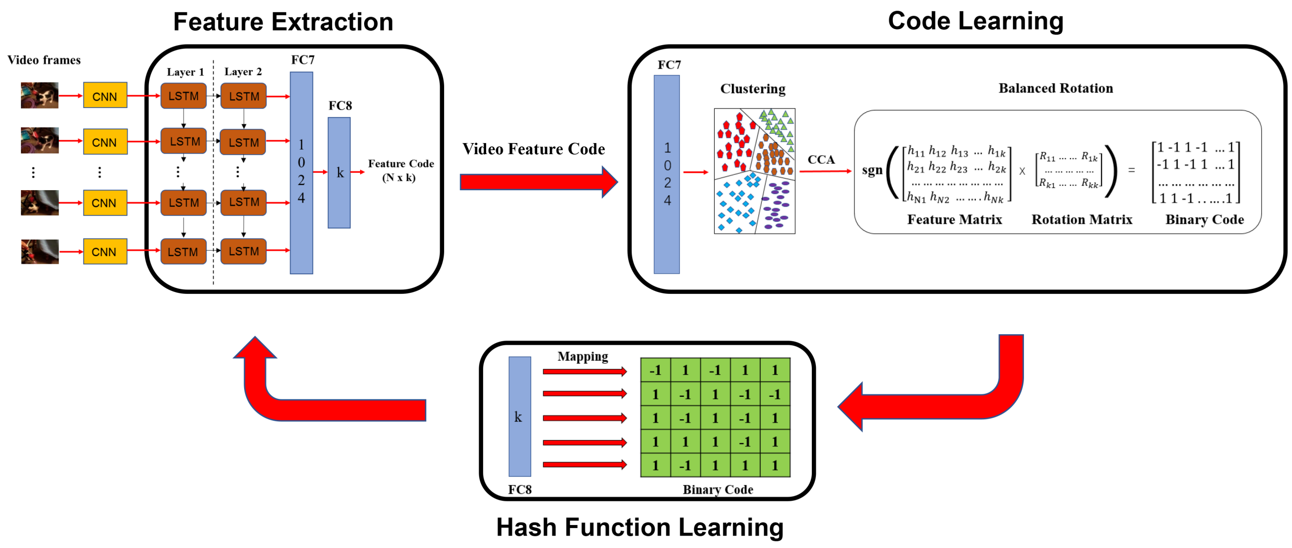

Figure 1: An overview of the proposed Unsupervised Deep Video Hashing for Content-based Video Retrieval. The whole framework is composed of three parts: feature extraction, code learning and hash function learning. The red arrows represent the direction of each iteration.

equivalently and allocated with the same number of bits in the quantization step afterwards. That is to say, directly applying image binarization to video features is unable to achieve effective hash codes for video retrieval. To address the above problem, we propose an end-to-end hashing framework (Figure 1), namely Unsupervised Deep Video Hashing (UDVH), with the aim to generate effective hash codes for fast retrieval. Our work differs from the existing methods in two aspects:

1. We propose an unsupervised deep hash framework and organize learning in a self-taught manner. Instead of minimizing feature reconstruction distortion [Zhang *et al.*, 2016], our framework minimizes the quantization error of projecting video features to a binary hypercube, allowing the feature extraction and hash function learning to engage with each other. Involving the feature clustering in the code learning enables the neighborhood structure to be preserved. To solve the objective function, which is a non-convex problem, an alternate approach is proposed, where the rotation matrix, binary code generation and the deep framework parameters are jointly optimized.
2. During the code learning, balanced rotation specifically designed for video features is proposed to find a proper projection matrix such that the variance of each projected dimension can be balanced. The balanced variance is maintained in each dimension, thereby facilitating the quantization step afterwards. The experimental results reveal that this improved binarization is suitable for hashing video features.

2 Methodology

In this section, we will introduce the main algorithmic modules involved in the proposed framework. First of all, some notations are defined here for ease of explanation. Assuming that there are N videos in the training set and each video contains m frames. In UDVH, VGG-19 [Simonyan and Zisserman, 2014] is used to extract the 4096-d feature vectors for all frames. The input feature matrix is denoted as $\mathbf{Z} \in \mathbb{R}^{4096\times(N\times m)}$ and it remains fixed in UDVH. Our goal is to develop the deep hash function that encodes $\mathbf{Z}$ into k-bit binary codes $b_N \in \{-1,+1\}^{N\times k}$, where $k << 4096$. We further denote the video feature matrix from fc7 layer as $\overline{\mathbf{Z}} \in \mathbb{R}^{N\times 1024}$. In the code learning stage, we first use K-means to categorize the training videos and then reduce the feature dimensionality by Canonical Correlation Analysis (CCA) [Hardoon *et al.*, 2004]. We denote the video feature matrix after dimensionality reduction as $\mathbf{H} \in \mathbb{R}^{N\times k}$, where k is the targeted bit size. Finally, with the proposed rotation matrix $\mathbf{R} \in \mathbb{R}^{k\times k}$, the binary code $\mathbf{B} \in \{-1,+1\}^{N\times k}$ can be generated to guide the hash function learning while iteratively updating the network parameters. In the following subsections, the proposed modules in UDVH will be elaborated.

2.1 Deep Hash Function

In this section, the deep hash function is defined along with a brief introduction about LSTM, which has been widely deployed in the sequence study for its powerful ability in dealing with the sequential input. The output of LSTM is actually depended on the previous state and the current input. Compared with the traditional Recurrent Neural Network (RNN) [Chung *et al.*, 2014], LSTM overcomes the problems of vanishing and exploding gradient by adding the memory cell, which is designed to store information such that the long-range temporal relationship could be exploited [Srivastava *et al.*, 2015]. Typically, a 2-layer LSTM unit is to tackle the frame-level features for the temporal-awareness. The frame-level features of training videos are encoded by the LSTM units sequentially and the encoded frame features are averagely fused by the connected fc7 layer to generate the video feature matrix $\overline{\mathbf{Z}}$. Thus, a robust representation for videos can be achieved taking both spatial and temporal information into account. We further denote the **deep hash function** as $\mathbf{F}(\mathbf{Z};\theta)$, where θ represents the parameters from LSTM and fully-connected layers for the concise description. Those parameters will be updated automatically during the process of hash function learning, thus producing the desirable output.

2.2 Feature Embedding

At the beginning stage of code learning, the video feature matrix $\overline{\mathbf{Z}}$ obtained from fc7 layer is clustered by K-means with the intention to classify the similiar videos into the same category, where the clustering parameter K is randomly selected. Such a clustering process is repeated in each loop for the more precise categorization. After K-means, the training videos are represented by their feature vectors and the corresponding fake labels ($\{0,1\}^{N\times k}$). They provide dedicated supervisory information for the following binarization such that the similar videos would be represented by the similar binary codes. Subsequently, CCA is adopted to reduce the dimensionality of video feature into the required bit size as the projected video feature matrix $\mathbf{H}$, where the correlation between features and labels is maximized and well preserved in the low-dimensional space. In contrast to PCA, CCA is more effective in extracting discriminative information with the robustness in anti-noise [Gong and Lazebnik, 2011], thus obtaining more robust video features after the projection. We denote the projection matrix as $\mathbf{P}\in\mathbb{R}^{1024\times k}$, then the process of dimensionality reduction can be simplified as:

$$\mathbf{H}=\overline{\mathbf{Z}}\times\mathbf{P},\ \text{s.t.}\ \overline{\mathbf{Z}}\in\mathbb{R}^{N\times 1024}. \tag{1}$$

However, the projection concentrates the most information in a few top eigenvectors, which will lead to imbalanced data and may lower the hash code quality [Xu *et al.*, 2013]. In the next section, we show the problem can be addressed by a novel rotation matrix.

2.3 Balanced Rotation

As mentioned previously, video features, compared to the image features, are likely to be more scattered in terms of distribution, leading to larger variance difference among projected dimensions. Usually, the quantization step after the projection treats each dimension equally and allocates the same number of bits to each dimension. That means the dimensions with low variance containing less information will have the same impact on Hamming distance calculation as the high variance dimensions containing more information. The hash codes generated by this scheme deemed ineffective. To address this problem, a balanced rotation matrix is required such that the information of each dimension is balanced and some important properties can be maintained. Firstly, the effect of rotation on the data variance is investigated, assuming the video features are zero centralized, i.e., $\sum_{i=1}^{N}\overline{\mathbf{Z}}_i=0$. Given the optimal projection $\mathbf{P}$, the projected data $\mathbf{H}=\overline{\mathbf{Z}}\mathbf{P}\in\mathbb{R}^{N\times k}$ is obtained by Eq.(1). Since $\overline{\mathbf{Z}}$ is centralized, the variance for the j-th dimension is $v_j=\sum_{i=1}^{N}\mathbf{H}_{ij}^2/N$. Given an orthogonal rotation matrix $\mathbf{R}$ and the adjusted data $\mathbf{H}_a=\mathbf{HR}$, the new variance of dimension v_j can be obtained. The sum of variance of each dimension can be calculated by:

$$\begin{aligned}N\sum_{j=1}^{k}{v_j}'&=\sum_{j=1}^{k}\sum_{i=1}^{N}(\mathbf{HR})_{ij}^2=tr(\mathbf{HRR}^T\mathbf{H}^T)\\&=tr(\mathbf{HH}^T)=\sum_{j=1}^{k}\sum_{i=1}^{N}(\mathbf{H})_{ij}^2=N\sum_{j=1}^{k}v_j.\end{aligned} \tag{2}$$

Eq. (2) indicates that the sum of variance on all dimensions in the centralized data is invariant after rotation. Our goal is to find a rotation such that the variance of each dimension can be balanced. Specifically, the degree of balance can be measured by the variance of standard deviation (VSD) of each dimension. Denote the standard deviation (SD) of the j-th dimension as $s_j={v_j}^{0.5}$, VSD is computed as:

$$\text{VSD}=\frac{1}{k}\sum_{i=1}^{k}(s_i-\overline{s})^2. \tag{3}$$

Theoretically, smaller VSD implies more balanced data. Hence, the goal can be achieved by finding a rotation which minimizes VSD. Directly minimizing Eq. (3) is, however, not intuitive. Based on the above analysis, we have $\sum_{i=1}^{k}s_i^2=\sum_{i=1}^{k}v_i=c$, where c is a constant. Therefore, the VSD in Eq. (3) can be further deduced as:

$$\begin{aligned}\text{VSD}&=\frac{1}{k}\sum_{i=1}^{k}(s_i-\overline{s})^2=\frac{1}{k}\sum_{i=1}^{k}s_i^2+\overline{s}^2-\frac{2}{k}\sum_{i=1}^{k}s_i\overline{s}\\&=\frac{c}{k}-\overline{s}^2=\frac{c}{k}-(\frac{1}{k}\sum_{i=1}^{k}s_i)^2=\frac{c}{k}-\frac{1}{k^2}\text{SSD}^2.\end{aligned} \tag{4}$$

So, it is straightforward to validate the following equivalence,

$$\min\ \text{VSD}\Leftrightarrow\min-\frac{1}{k^2}\text{SSD}^2\Leftrightarrow\max\ \text{SSD}. \tag{5}$$

This indicates that minimizing the VSD of data by a rotation is equivalent to maximizing the sum of standard deviation (SSD). Furthermore, the SSD can be written in the matrix form as:

$$\text{SSD}=\sum_{j=1}^{k}(\frac{1}{N}\sum_{i=1}^{N}\mathbf{H}_{ij}^2)^{0.5}=\frac{1}{\sqrt{N}}\|\mathbf{H}^T\|_{2,1}, \tag{6}$$

where $\|.\|_{2,1}$ is the $l_{2,1}$-norm of matrix. Now, with Eq. (5) and Eq. (6), the rotation matrix which can balance the variance of each dimension can be learned from the following optimization formulation,

$$\max_{\mathbf{R}}\|\mathbf{R}^T\mathbf{H}^T\|_{2,1},\ \text{s.t.}\ \mathbf{RR}^T=\mathbf{I}_k, \tag{7}$$

where $\mathbf{I}_k$ is the identity matrix. Eq. (7) is also the objective function of balanced rotation.

2.4 Learning Objective

In this section, we introduce the objective function of UDVH in details. There are two terms involved in defining the objective function. First, we use the binary code $\mathbf{B}$ as the guidance in the hash function learning, where the Euclidean distance is minimized between the output of top layer (fc8) in the deep network and $\mathbf{B}$. It can be formulated as:

$$\min_{\theta,\mathbf{R}} \|\mathbf{F}(\mathbf{Z};\theta) - \mathbf{B}\|_F^2, \text{ s.t. } \mathbf{B} \in \{-1,+1\}^{N\times k}. \tag{8}$$

Considering also the balanced rotation, the final objective function of UDVH will be

$$\begin{aligned} &\min_{\theta,\mathbf{B},\mathbf{R}} \|\mathbf{F}(\mathbf{Z};\theta) - \mathbf{B}\|_F^2 - \lambda\|\mathbf{R}^T\mathbf{H}^T\|_{2,1}, \\ &\text{s.t. } \mathbf{B} \in \{-1,+1\}^{N\times k},\ \mathbf{R}\mathbf{R}^T = \mathbf{I}_k, \end{aligned} \tag{9}$$

where λ is the balance parameter. However, it is not possible to solve Eq. (9) directly due to the binary constraints.

3 Optimization

Since it is a non-convex problem with the binary constraints in Eq. (9), an alternate approach is proposed to optimize the objective, where the deep hash function $\mathbf{F}(\overline{\mathbf{Z}};\theta)$ can be learned by iteratively updating the parameters, such as θ, $\mathbf{B}$ and $\mathbf{R}$, during each loop. Usually it takes just a few iterations to complete the optimization process for one specific bit size. Initially, a random value is specified to θ and the input matrix $\mathbf{Z}$ is encoded by the initialized deep network. Conducting clustering and dimensionality reduction can obtain the projected video features. Afterwards, we carry out the following three steps to accomplish the goal.

Update R. With other parameters fixed, problem (9) is reduced to the orthogonality constrained $l_{2,1}$-norm maximization problem in Eq. (7), which is defined as:

$$\begin{aligned} &\min_{\mathbf{R}} \|-\mathbf{R}^T\mathbf{H}^T\|_{2,1} = \max_{\mathbf{R}} \|\mathbf{R}^T\mathbf{H}^T\|_{2,1}, \\ &\text{s.t. } \mathbf{R}\mathbf{R}^T = \mathbf{I}_k,\ \mathbf{H} \in \mathbb{R}^{N\times k}. \end{aligned} \tag{10}$$

The above problem can be efficiently solved by the gradient flow method [Wen and Yin, 2013]. Specifically, we first define $M_k = \{\mathbf{R} \in \mathbb{R}^{k\times k} : \mathbf{R}\mathbf{R}^T = \mathbf{I}_k\}$ as the feasible set, which is also called the Stiefel manifold. Then, the tangent space for M_k is $T_R M_k = \{\mathbf{T} \in \mathbb{R}^{k\times k} : \mathbf{T}^T\mathbf{R} + \mathbf{R}^T\mathbf{T} = 0\}$. Here, the basic idea is to find an optimal direction in the tangent space of current point $\mathbf{R}$ like conventional gradient descent, then project the direction to the feasible manifold, and replace the current point with the projected one. Iterating such steps and finally it will arrive at a stationary point. To optimize problem (10), the convergence theorem is proposed and illustrated as: given a point $\mathbf{R}_t$ in the feasible set, the below updating rule will lead to larger value unless it has arrived at a stationary point,

$$\mathbf{R}_{t+1} = \mathbf{Q}_t\mathbf{R}_t, \tag{11}$$

where $\mathbf{Q}_t$ is the Cayley transformation matrix defined by

$$\mathbf{Q}_t = (\mathbf{I}_k + \frac{\tau}{2}\mathbf{W}_t)^{-1}(\mathbf{I}_k - \frac{\tau}{2}\mathbf{W}_t), \tag{12}$$

$$\mathbf{W}_t = \mathbf{G}_t\mathbf{R}_t^T - \mathbf{R}_t\mathbf{G}_t^T, \tag{13}$$

$$\mathbf{G}_t = -\mathbf{H}_t^T\mathbf{H}\mathbf{R}_t\mathbf{D}_t, \tag{14}$$

$$\mathbf{D}_t = diag(\frac{1}{N^{0.5}s_{t1}}, ..., \frac{1}{N^{0.5}s_{ti}}, ..., \frac{1}{N^{0.5}s_{tk}}). \tag{15}$$

Here, the subscript t denotes the t-th iteration, τ is a step size satisfying Armijo-Wolfe conditions [Nocedal and Wright, 2006], and s_{ti} represents the standard deviation of $(\mathbf{H}\mathbf{R}_t)_{*i}$. The above steps are iteratively executed until convergence and the obtained $\mathbf{R}$ is the rotation matrix.

Update B. By fixing $\mathbf{Z}$, $\mathbf{H}$ and $\mathbf{R}$, Eq. (9) has been reduced to:

$$\min_{\theta,\mathbf{B}} \|\mathbf{F}(\mathbf{Z};\theta) - \mathbf{B}\|_F^2, \text{ s.t. } \mathbf{B} \in \{-1,+1\}^{N\times k}. \tag{16}$$

Since we have obtained the rotation matrix $\mathbf{R}$ in the previous step and the binary code $\mathbf{B}$ can be directly computed as:

$$\begin{aligned} &\mathbf{B} = sgn(\mathbf{H} \times \mathbf{R}), \\ &\text{s.t. } \mathbf{H} \in \mathbb{R}^{N\times k},\ \mathbf{R} \in \mathbb{R}^{k\times k}, \end{aligned} \tag{17}$$

where $sgn(x) = 1$ if $x > 0$ and -1 otherwise.

Update θ. With fixed $\mathbf{Z}$, $\mathbf{R}$ and $\mathbf{B}$, Eq. (9) is reduced to

$$\min_{\theta} \|\mathbf{F}(\mathbf{Z};\theta) - \mathbf{B}\|_F^2, \text{ s.t. } \mathbf{B} \in \{-1,+1\}^{N\times k}. \tag{18}$$

This minimization problem can be solved by fine-tuning the deep network with Stochastic Gradient Descent (SGD) until it gets converged, where the Euclidean loss is minimized via mini-batch back-propagation and the low bound can be found. The network parameter θ is updated simultaneously and determined in the current loop after convergence. Then, the deep network can produce new video features for the next loop. Those parameters are updated sequentially for 10 loops and finally the deep hash function can be built. Given a query video $\mathbf{Z}_q$, we can create its hash codes with $sgn(\mathbf{F}(\mathbf{Z}_q;\theta))$. The overall UDVH is summarized in Algorithm 1.

Algorithm 1: Unsupervised Deep Video Hashing

```
Input: The input feature matrix $\mathbf{Z}$; Randomly initialize deep
    parameters $\theta$;
Output: $\mathbf{F}(\mathbf{Z};\theta)$: deep hash function;
 1: for $t = 1$ to 10 do
 2:     Obtain projected video feature matrix $\mathbf{H}$;
 3:     Update rotation matrix $\mathbf{R}_t$ according to Eq.
        (11)~(15);
 4:     Update binary code $\mathbf{B}_t$ according to Eq. (17);
 5:     Update deep parameters $\theta$ according to Eq. (18);
 6:     Until Convergence;
 7: end for
 8: return $\mathbf{F}(\mathbf{Z};\theta)$;
```

4 Experiment

4.1 Datasets and Experimental Settings

In this experimental study, two large-scale video datasets are adopted, which contain:

FCVID [Over *et al.*, 2014]: There are 91,223 videos collected from Youtube in the dataset, which are classified into

239 categories. It covers a wide range of popular topics, such as animals, events and various scenes. In the training phase, we randomly select 45,611 videos for the train split and the rest is used as the test split. In the retrieval phase, 1,000 videos randomly picked up from the test slip are used as the query and others form the gallery.

YFCC [Thomee *et al.*, 2015]: It is the largest public video dataset, containing over 0.8M videos. Due to the invalid urls and the corrupted video files, about 0.7M videos have been processed in the experiment, where 0.1M labeled videos [Zhang *et al.*, 2016] construct the test split in the retrieval while 0.6M unlabeled videos form the train split in the unsupervised learning. In the labeled split, there are 80 categories collected from the third level of MIT SUN scene hierarchy [Xiao *et al.*, 2010]. For the fair comparison, we select 1,000 videos randomly as the query and 99,000 videos as the gallery. For each video, 20 frames are uniformly selected as the representation of the video.

The algorithm is implemented under the open-source Caffe framework [Jia *et al.*, 2014]. The server configurations are: Intel Core i7-5960X CPU, 64GB RAM and a TITAN X GPU.

4.2 Baselines and Evaluation Metrics

Several recent hashing methods are adopted as baselines in the experiment, which are Deep Hashing (DH) [Erin Liong *et al.*, 2015], Iterative Quantization (ITQ) [Gong and Lazebnik, 2011], Submodular video hashing (SubMod) [Cao *et al.*, 2012], Spectral Hashing (SP) [Weiss *et al.*, 2009], Multiple Feature Hashing (MFH) [Song *et al.*, 2011], Anchor Graph Hashing (AGH) [Liu *et al.*, 2011] and Self-Supervised Temporal Hashing (SSTH) [Zhang *et al.*, 2016]. Identical train and test sets are organized for all methods and the best performance is tuned based on the data settings in their papers. Regarding the evaluation, mean average precision at top K retrieved videos (mAP@K) [Over *et al.*, 2014] is adopted as the main metric. Precision-Recall curve and HD2 precision are two additional ones [Davis and Goadrich, 2006].

4.3 Results and Discussions

Architecture Investigation

In the proposed framework, the number of clusters during the clustering is the only hyper parameter. Figure 2(a) shows the effect on mAP@20 when varying the cluster numbers. Due to the space limit, only the results of 64 bits and 128 bits are reported. Seen from the figure, the mAP value increases dramatically with the bigger cluster number selected at the beginning and remains stable over 400 categories. This indicates the cluster number has tremendous influence on the system performance. The sophisticated clustering encourages the similar videos to be classified into the same category such that similar hash codes can be generated for those videos. For both datasets, it seems that the performance get saturated when the cluster number reaches 400, meaning this parameter is easy to set. Additionally, the Euclidean loss during the hash function learning on FCVID is plotted in Figure 2(b). Clearly, fast convergence (after about 4 loops) can be achieved because of the dedicated LSTM units included in the proposed hashing framework.

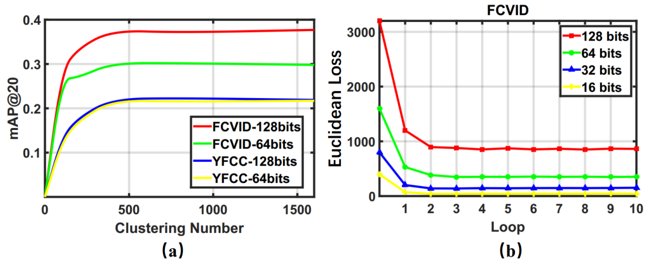

Figure 2: (a) The mAP@20 of various cluster numbers on both datasets when using 64 bits and 128 bits; (b) The Euclidean Loss after the hash function learning in each loop.

Table 1: The mAP@K of 64 bits and 128 bits when using CCA+ITQ, PCA+BR and CCA+BR respectively in the code learning process on both datasets.

Dataset	FCVID											
Bits	64 bits						128 bits					
mAP@K	K=5	K=20	K=40	K=60	K=80	K=100	K=5	K=20	K=40	K=60	K=80	K=100
PCA+BR	0.209	0.136	0.105	0.09	0.08	0.073	0.358	0.302	0.283	0.274	0.266	0.258
CCA+BR	**0.33**	**0.302**	**0.283**	**0.274**	**0.266**	**0.258**	**0.412**	**0.363**	**0.354**	**0.34**	**0.329**	**0.319**

Dataset	YFCC											
Bits	64 bits						128 bits					
mAP@K	K=5	K=20	K=40	K=60	K=80	K=100	K=5	K=20	K=40	K=60	K=80	K=100
CCA+ITQ	0.21	0.192	0.175	0.152	0.141	0.129	0.219	0.198	0.175	0.159	0.146	0.135
CCA+BR	**0.243**	**0.218**	**0.183**	**0.165**	**0.15**	**0.136**	**0.256**	**0.221**	**0.184**	**0.159**	**0.149**	**0.135**

Next, the retrieval results are evaluated by using various hashing strategies adopted in the phase of code learning in order to validate the claimed contributions. Three combinations are included: CCA+ITQ, PCA+BR and CCA+BR, as illustrated in Table 1. The results show combination of CCA and BR substantially outperforms the other two methods on the datasets. Compared with the PCA method, more valuable information is concentrated in the top eigenvectors by CCA after the dimensionality reduction because of the dedicated supervisory information (predicted labels) created by the clustering. It turns out the choice of CCA, rather than PCA, is sensible. Moreover, the comparison results of CCA+ITQ against CCA+BR clearly demonstrate that the balanced hash codes help to improve the retrieval performance.

Comparison with State-of-The-Arts

We also compare our UDVH with the baselines, given a retrieval task. Figure 3 shows the mAP@K results of various bit sizes on both datasets when using different methods. As can be seen, the proposed method outperforms the state-of-the-art hashing approaches significantly. Specifically, the mAP@5 value of our algorithm is 41.2% when using 128-bit code on FCVID, which is about 6% higher than the results achieved by the most comparable SSTH. The mAP curves are even more stable than the other methods with the increasing returned video number. When testing the algorithms on YFCC dataset, the mAP values of all algorithms drop down, compared to the results on FCVID using the same bits. The gap between the mAP@5 values of UDVH and SSTH is reduced trivially to around 4%, which is still big. The reason behind is that most videos in YFCC are taken by mobile phones from the amateurs instead of the professionals in FCVID [Thomee

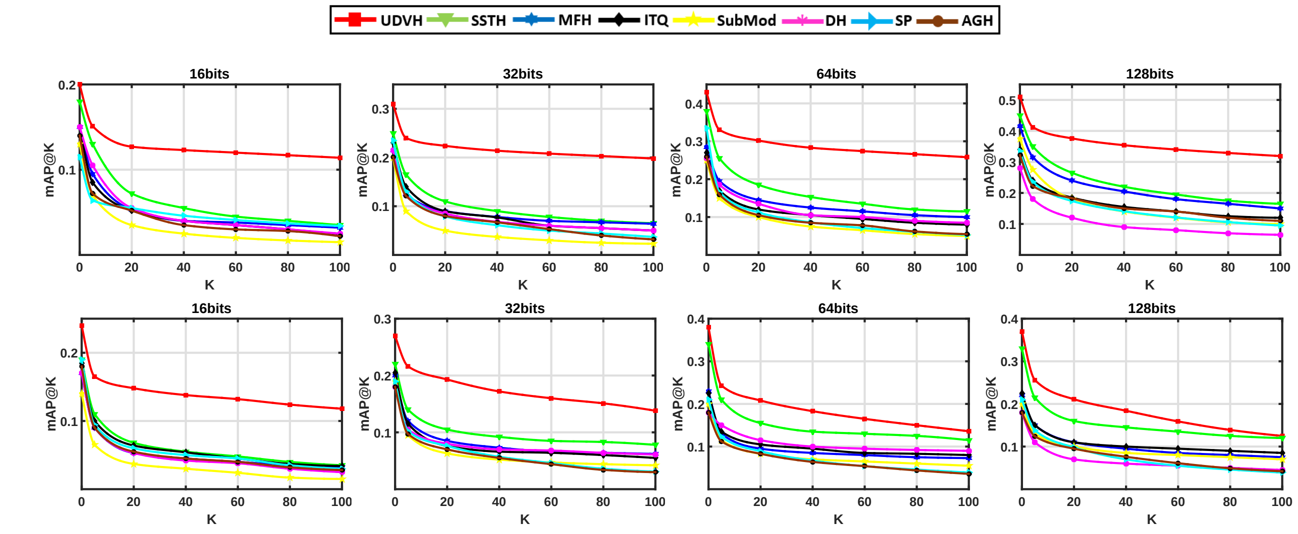

Figure 3: The mAP@K of various video hashing methods at different bit sizes. Top: FCVID; Bottom: YFCC.

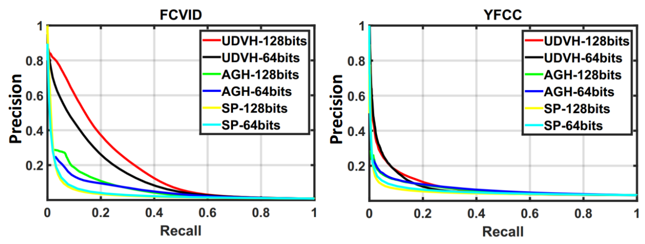

Figure 4: Precision-Recall curves when using 64 bits and 128 bits on both datasets. Left: FCVID; Right: YFCC.

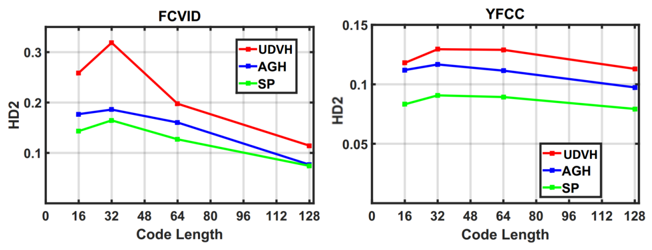

Figure 5: HD2 precision at various bit sizes on both datasets. Left: FCVID; Right: YFCC.

et al., 2015]. The video retrieval has become a more challenging task for the low-quality videos when using YFCC dataset. Figure 4 and Figure 5 respectively show the Precision-Recall curves and HD2 precision curves at various bit sizes on both datasets, in which the ranking number is 100 and the Hamming radius is 2. Compared with the results from several conventional hashing methods, the best performance is again achieved by UDVH on both evaluation metrics.

In Figure 6, the detailed performance of the mAP@5 achieved by two methods, UDVH and SSTH, is illustrated when using 128 bits. Due to the limited space, only the results on FCVID are reported here. Given three query videos, our UDVH correctly finds five similar videos for each of them whereas SSTH makes mistakes in recognizing swimming amateur and dolphin. The major reason is that our UDVH tends

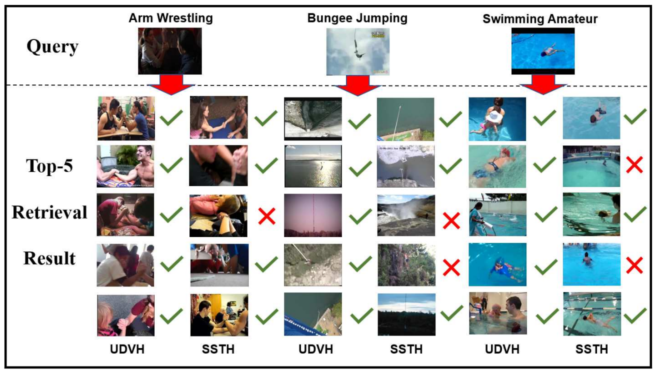

Figure 6: The retrieval results of 128 bits when using SSTH and UDVH on FCVID.

to preserve the neighborhood structure by incorporating the clustering into the hash code learning, which is crucial for a nearest neighbour search task like this.

5 Conclusion

In this paper, a novel unsupervised video hashing framework called Unsupervised Deep Video Hashing has been proposed to address the existing problems in the large-scale video retrieval. Compared with the conventional video hashing methods, the proposed approach is clearly superior because the feature extraction, code learning and hash function learning are combined jointly in a self-taught fashion. Moreover, we applied a balanced rotation to the video-specific features such that the variance of dimensions when projecting them into low-dimensional space can be balanced, which helps to generate effective video hash codes. The effectiveness of proposed framework is demonstrated based on the superior retrieval results from two video datasets.

References

[Cao *et al.*, 2012] Liangliang Cao, Zhenguo Li, Yadong Mu, and Shih-Fu Chang. Submodular video hashing: a unified framework towards video pooling and indexing. In *ACM Multimedia*, pages 299–308. ACM, 2012.

[Chung *et al.*, 2014] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[Davis and Goadrich, 2006] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *ICML*, pages 233–240. ACM, 2006.

[Erin Liong *et al.*, 2015] Venice Erin Liong, Jiwen Lu, Gang Wang, Pierre Moulin, and Jie Zhou. Deep hashing for compact binary codes learning. In *CVPR*, pages 2475–2483, 2015.

[Gong and Lazebnik, 2011] Yunchao Gong and Svetlana Lazebnik. Comparing data-dependent and data-independent embeddings for classification and ranking of internet images. In *CVPR*, pages 2633–2640. IEEE, 2011.

[Hardoon *et al.*, 2004] David R Hardoon, Sandor Szedmak, and John Shawe-Taylor. Canonical correlation analysis: An overview with application to learning methods. *Neural computation*, 16(12):2639–2664, 2004.

[Jia *et al.*, 2014] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, pages 675–678. ACM, 2014.

[Karpathy *et al.*, 2016] Andrej Karpathy, Justin Johnson, and Fei-fei Li. Visualizing and understanding recurrent networks. In *Proceedings of ICLR*, 2016.

[Liong *et al.*, 2016] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan, and Jie Zhou. Deep video hashing. *IEEE Transactions on Multimedia*, 2016.

[Liu *et al.*, 2011] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.

[Nocedal and Wright, 2006] Jorge Nocedal and Stephen J Wright. *Sequential quadratic programming*. Springer, 2006.

[Over *et al.*, 2014] Paul Over, Jon Fiscus, Greg Sanders, David Joy, Martial Michel, George Awad, Alan Smeaton, Wessel Kraaij, and Georges Quénot. Trecvid 2014–an overview of the goals, tasks, data, evaluation mechanisms and metrics. In *Proceedings of TRECVID*, page 52, 2014.

[Simonyan and Zisserman, 2014] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[Song *et al.*, 2011] Jingkuan Song, Yi Yang, Zi Huang, Heng Tao Shen, and Richang Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM Multimedia*, pages 423–432. ACM, 2011.

[Srivastava *et al.*, 2015] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhutdinov. Unsupervised learning of video representations using lstms. In *ICML*, pages 843–852, 2015.

[Thomee *et al.*, 2015] Bart Thomee, David A Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 1(8), 2015.

[Wang and Schmid, 2013] Heng Wang and Cordelia Schmid. Action recognition with improved trajectories. In *ICCV*, pages 3551–3558, 2013.

[Weiss *et al.*, 2009] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2009.

[Wen and Yin, 2013] Zaiwen Wen and Wotao Yin. A feasible method for optimization with orthogonality constraints. *Mathematical Programming*, 142(1-2):397–434, 2013.

[Xiao *et al.*, 2010] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, pages 3485–3492. IEEE, 2010.

[Xu *et al.*, 2013] Bin Xu, Jiajun Bu, Yue Lin, Chun Chen, Xiaofei He, and Deng Cai. Harmonious hashing. In *IJCAI*, 2013.

[Ye *et al.*, 2013] Guangnan Ye, Dong Liu, Jun Wang, and Shih-Fu Chang. Large-scale video hashing via structure learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2272–2279, 2013.

[Zhang *et al.*, 2016] Hanwang Zhang, Meng Wang, Richang Hong, and Tat-Seng Chua. Play and rewind: Optimizing binary representations of videos by self-supervised temporal hashing. In *ACM Multimedia*, pages 781–790. ACM, 2016.