This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2016.2645404, IEEE Transactions on Multimedia

Deep Video Hashing

Venice Erin Liong, *Student Member, IEEE*, Jiwen Lu, *Senior Member, IEEE*, and Yap Peng Tan, *Senior Member, IEEE*

***Abstract*—In this work, we propose a new deep video hashing (DVH) method for scalable video search. Unlike most existing video hashing methods which first extract features for each single frame and then use conventional image hashing techniques, our DVH learns binary codes for the entire video with a deep learning framework so that both temporal and discriminative information can be well exploited. Specifically, we fuse the temporal information across different frames within each video to learn the feature representation under two criteria: 1) the distance between a feature pair obtained at the top layer is small if they are in the same class, and large if they are from different classes, and 2) the quantization loss between the real-valued features and the binary codes is minimized. We exploit different deep architectures to utilize spatial-temporal information in different manners and compare them with single frame based deep feature models and state-of-the-art image hashing algorithms. Experimental results on two video databases demonstrate the effectiveness of our proposed method.**

***Index Terms*—Scalable video search, video hashing, deep learning.**

I. Introduction

Fast and efficient visual search is a challenging task which has gained large interests in computer vision, especially with the increasing amount of multimedia content which are available over the internet in recent years. This task aims to retrieve the most relevant visual content from a database, given a query sample, in an accurate and efficient manner. In contrast to visual images, videos provide diverse and complex visual patterns consisting of low-level visual content in each frame as well as high-level structured content, such as actions or events, across frames [6], [20], [21], [30], [63]. This makes video search more challenging than image search. Moreover, each video may have a number of image frames which leads to exhaustive computation and comparisons between frames, which is impractical when we have a large video database. As can be seen in surveillance and social media (YouTube), videos are equally important as images. Hence, how to develop a framework for scalable video search where discriminative information from videos can be well exploited in the extracted features remains an important problem in visual search, especially considering that discriminative information is preserved as much as possible with minimal computation cost and memory storage.

A key topic in visual search is *learning-based hashing*, which transforms high-dimensional feature vectors to compact binary codes, by preserving visual content using statistical inference. However, most current works in scalable visual search focus on image-based retrieval [16], [17], [33], [35], [38], [41], [42], [69] or text-image/image-text retrieval [37], [44], [58], [59], [68]. To our best knowledge, there are only few works that present an efficient framework for video hashing. Hence, it is desirable to make better use of hashing methods to exploit the spatial-temporal information of videos.

Current video hashing methods are mainly applied in two multimedia computing applications. The first is the near-duplicate search where conventional hashing techniques [12], [25], [57] were used to identify duplicate videos efficiently [7], [8], [52]. The second is content-based video retrieval [2], [35], [61], [63] which retrieves the most semantically similar videos from a database for a given query video. In this work, we focus on the latter one. Video hashing for content-based retrieval can be divided into three categories. The first extracts a single representative feature vector, and then performs hashing. The second treats each frame as an image and performs image hashing first such that the resulting frame-frame hamming distances of two videos are then combined through averaging. The last selects several representative frames and employs image hashing on these selected frames. While these frameworks are straight-forward, they cannot exploit the temporal information of videos in the learning stage. Furthermore, hashing image frames individually is computationally expensive considering that each video would usually have a minimum of 30-50 frames. Hence, it is desirable to present a video hashing framework which learns strong features by utilizing the temporal information, and in turn, also minimizes the hamming distance computation.

To address these challenges, we propose a Deep Video Hashing (DVH) method for scalable video search where we explore different CNN-based architectures. Since Convolutional Neural Networks (CNNs) have shown promising performance in several computer vision tasks due to their strong feature representation capability, we also employ CNN in our video hashing framework. The basic idea of our approach is shown in Fig. 1. Specifically, we build an end-to-end deep CNN learning framework which utilizes spatio-temporal information after the stacked convolutional-

Venice Erin Liong is with the Interdisciplinary Graduate School, Rapid-Rich Object Search (ROSE) Lab, Nanyang Technological University, Singapore, 639798. E-mail: veniceer001@e.ntu.edu.sg.

Jiwen Lu is with the Department of Automation, Tsinghua University, State Key Lab of Intelligent Technologies and Systems, and Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing, 100084, China. E-mail: lujiwen@tsinghua.edu.cn.

Yap-Peng Tan is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, 639798. Email: eyptan@ntu.edu.sg.

EDICS: 5-SEAR: Multimedia Search and Retrieval.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2016.2645404, IEEE Transactions on Multimedia

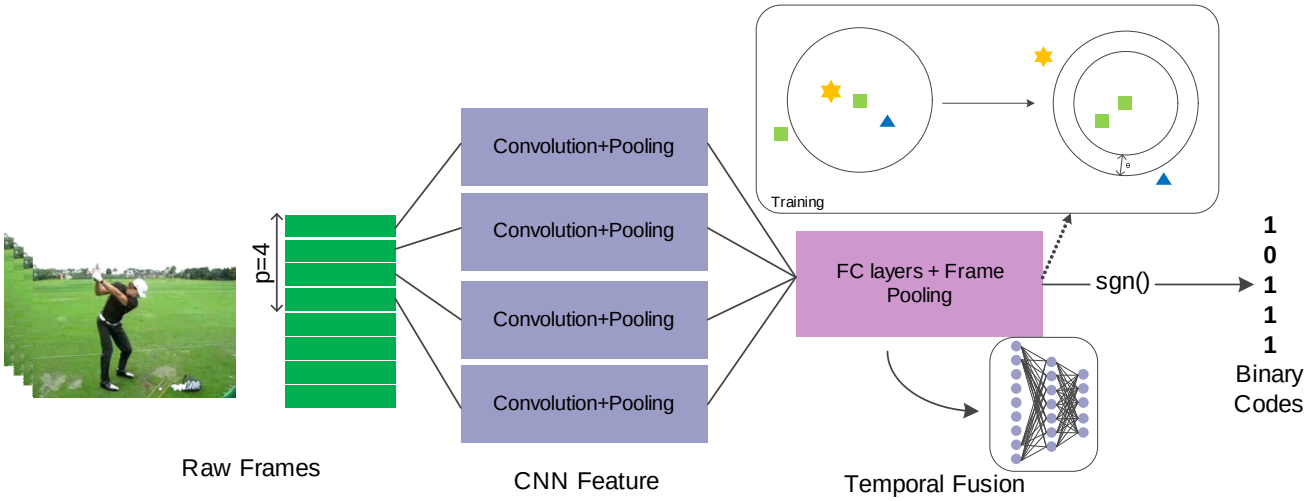


Fig. 1. Basic idea of our proposed DVH approach. We pass a set of successive image frames with a pre-defined frame number (in this figure, $p = 4$) to the convolutional and pooling layers to obtain frame-wise CNN feature representation. Then, we perform temporal fusion in the fully-connected layers. At the final stage of our deep network, we apply the $\text{sgn}(\cdot)$ in the activation outputs and obtain the compact binary codes. At the training stage, the network parameters are learned using back-propagation with a large-margin cost function such that video features that contain similar semantics/label information are close to each other, while video features that are dissimilar are far apart as possible, and a quantization loss criterion such that the real feature codes and binary codes are similar as much as possible.

pooling layers to extract representative video features, and obtain compact binary codes. We discriminatively train our model with a Siamese network, by maximizing the inter-class distance and minimizing the intra-class distance of the video feature pairs, as well as minimizing the quantization loss between real-value codes and binary codes. We exploit different spatial-temporal feature pooling architectures and compare them to single-frame CNN architectures as well as state-of-the-art image-based hashing methods. Experimental results on two video datasets show the effectiveness of our proposed approach.

The contributions of this work are summarized as follows:

1) We propose a deep learning-based hashing method called deep video hashing (DVH) which learns a deep network to exploit the discriminative and temporal information of videos in order to represent each video with meaningful binary codes.
2) We conduct extensive scalable video search experiments on two video datasets to demonstrate the efficacy of our proposed method. We exploit different frame fusion architectures and compare them to several baseline network architectures, state-of-the-art single frame hashing methods, and existing video hashing methods.

The rest of the paper is organized as follows. Section II briefly reviews related work. Sections III presents the proposed deep video hashing methods with different temporal fusion architectures. Section IV presents the experimental results, and Section V concludes the paper.

II. Related Work

In this section, we briefly review three related topics: 1) learning-based hashing, 2) video hashing, and 3) deep learning for video analysis.

A. Learning-based Hashing

Several learning-based hashing methods such as subspace models [16], [55], manifold models [22], [50], and kernel models [39], [45] have been exploited in the literature. These methods can be classified into two classes: *unsupervised* [9], [16], [17], [43], [57] and *supervised* [27], [36], [38], [39], [49], [50], [55]. The first category does not require label information. For example, Gong *et al.* [16] proposed a PCA-ITQ method which learns hashing functions by first performing PCA to maximize the variance of the hash bits and then learning a rotation matrix to minimize the quantization loss. Liu *et al.* [40] proposed an Anchor Graph Hashing (AGH) method by using the concept of *anchors* to identify the similarity between features. For the second category, class-wise label information is utilized. For example, Gong *et al.* [16] extended the PCA-ITQ to CCA-ITQ which first performs CCA to maximize the correlation between semantically similar features, and then minimizes the quantization loss. Liu *et al.* [39] employed the Kernel Supervised Hashing (KSH) method to perform kernel mapping and utilize supervised information through minimizing the distance of similar pairs and maximizing the distance of dissimilar pairs. Lin *et al.* [36] proposed a FastHash method to learn the binary codes through Graph Cuts and a greedy boosted decision tree framework. While these methods are mostly nonlinear hashing techniques, only a few works have used deep learning techniques to perform end-to-end nonlinear mapping [3], [10], [66], [67], [70]. Furthermore, these methods are specifically developed for large-scale image retrieval. In contrast, our work focuses on using statistical knowledge for scalable video search.

B. Video Hashing

Previous works on video hashing were mostly used for near-duplicate video retrieval tasks and several of them

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2016.2645404, IEEE Transactions on Multimedia

focused on video feature representation rather than learning the hashing functions [7], [8], [52], [56], [63]. For example, Song *et al.* [52] introduced a multiple feature hashing method which utilizes multiple features and extracts different local structures to obtain efficient binary codes. Douze *et al.* [8] extracted representative spatial-temporal features from images and used conventional hashing methods to obtain binary codes. There are only a few video hashing methods proposed for content-based retrieval. For example, Cao *et al.* [2] proposed a submodular hashing framework which selects relevant frames from videos to learn the hashing functions for efficient video search. However, it did not really learn a video-based hashing function by using statistical knowledge. Ye *et al.* [61] proposed a supervised structural learning framework which exploits the temporal consistency to learn the linear hashing functions. However, it only learns a linear projection which may not truly capture the nonlinear nature of video data. Li *et al.* [34], [35] proposed a hashing model across the Euclidean space and the Riemannian manifold, which learns hashing functions based on the kernel max-margin framework for face video retrieval. However, their work represented videos with a single feature representation (covariance matrix), which may not fully exploit the spatio-temporal information in videos.

C. Deep Learning for Video Analysis

Deep learning techniques have shown great success in various computer vision tasks such as in image recognition [31], [53], scene labeling [11], [13], and pedestrian detection [65]. While a number of deep learning methods have also been proposed for video analysis, most of them focused on video action recognition [1], [26], [46], [51], video classification [29], [62], [64] and event detection [48], [60]. For example, Ji *et al.* [26] introduced a 3D Convolution Neural Networks approach which considers spatial and temporal information for action recognition. Karpathy *et al.* [29] presented an extensive evaluation of different deep architectures for large-scale video classification where they introduced different spatio-temporal convolutions based on how the frames are fused in the network. Ng *et al.* [64] exploited different feature fusion methods after the stacked convolution and pooling layers and investigated the Long Short Term Memory (LSTM) networks for video classification. Xu *et al.* [60] explored pooling and encoding methods to combine frame-level features for event detection. Simonyan *et al.* [51] implemented a two stream convolution network for action recognition in videos to model the spatial and temporal data individually, and fuse the scores together. To our knowledge, nobody has investigated deep architectures for video hashing.

III. Proposed Approach

In this section, we first present the basic idea of the learning-based video hashing model, then describe our deep video hashing (DVH) method and its implementation details.

A. Learning-based Video Hashing

Let $\mathcal{X} = \{\mathbf{X}_i, y_i\}_{i=1}^{M}$ be a collection of M videos where $\mathbf{X}_i = [\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \ldots, \mathbf{x}_{i,f_i}] \in \mathbb{R}^{d \times f_i}$ is the ith video with successive f_i frames, y_i is the label information, and $\mathbf{x}_{ij} \in \mathbb{R}^d$ is the jth image feature frame of the video $\mathbf{X}_i$ with a feature length of d. The objective of learning-based video hashing is to learn K hash functions to project each video into a single or multiple K-bit compact binary vectors as follows:

$$\mathcal{F}_{\mathbf{X}_i} : \mathbb{R}^{d \times f_i} \rightarrow \{-1, 1\}^{K \times g_i} \tag{1}$$

where $g_i \in [1, f_i]$[1].

To obtain hashing functions, we learn a linear projection matrix, $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots \mathbf{w}_K] \in \mathbb{R}^{d \times K}$ to map the video features into compact binary codes. To obtain a single binary vector for the ith video, $\mathbf{b}_i \in \{-1, 1\}^{K \times 1}$, we first extract a compact single feature representation for the ith video defined as $\tilde{\mathbf{x}}_i$, and then project it linearly as follows:

$$\mathbf{b}_i = \text{sgn}(\mathbf{W}^\top \tilde{\mathbf{x}}_i) \tag{2}$$

However, it is difficult to represent a whole video as a single binary code without losing significant amount of information. Hence, a single video can be represented into multiple binary vectors by treating each frame as an image feature and performing image-based hashing. To obtain multiple binary vectors for the ith video, $\mathbf{B}_i \in \{-1, 1\}^{K \times f_i}$, we compute the frame-wise feature representation as follows:

$$\mathbf{B}_i = \text{sgn}(\mathbf{W}^\top \mathbf{X}_i) \tag{3}$$

These conventional learning-based video hashing methods make use of hand-crafted single representation features [34], [35] and/or use a single linear projection [61], which may not effectively capture the nonlinear relationship of video representations and cannot exploit temporal information present in videos.

B. Deep Video Hashing

Our work employs a deep learning model to learn several nonlinear projections to obtain compact binary codes, where both the discriminative and temporal information of videos are exploited in an end-to-end learning framework. By doing so, we are able to learn powerful video representations in a spatial-temporal level. Unlike previous video hashing methods which either learn hashing functions from a single video feature representation or frame-by-frame, we process a set of successive frames to obtain a single binary vector which leads to a fewer number ($g_i < f_i$) of binary codes to represent the ith video. Therefore, we are able to minimize the number of binary codes to represent each video but still extract significant information as much as possible.

As shown in Fig. 1, given a fixed frame size p, we have a set of image frames $\mathbf{I}_u \in \mathbb{R}^{p \times h \times w \times 3}$ that is passed through a series of convolution and pooling layers with

[1]During learning, the binary codes are set to {-1,1} to ensure proper centering, but they can then be set to {0,1} during retrieval.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2016.2645404, IEEE Transactions on Multimedia

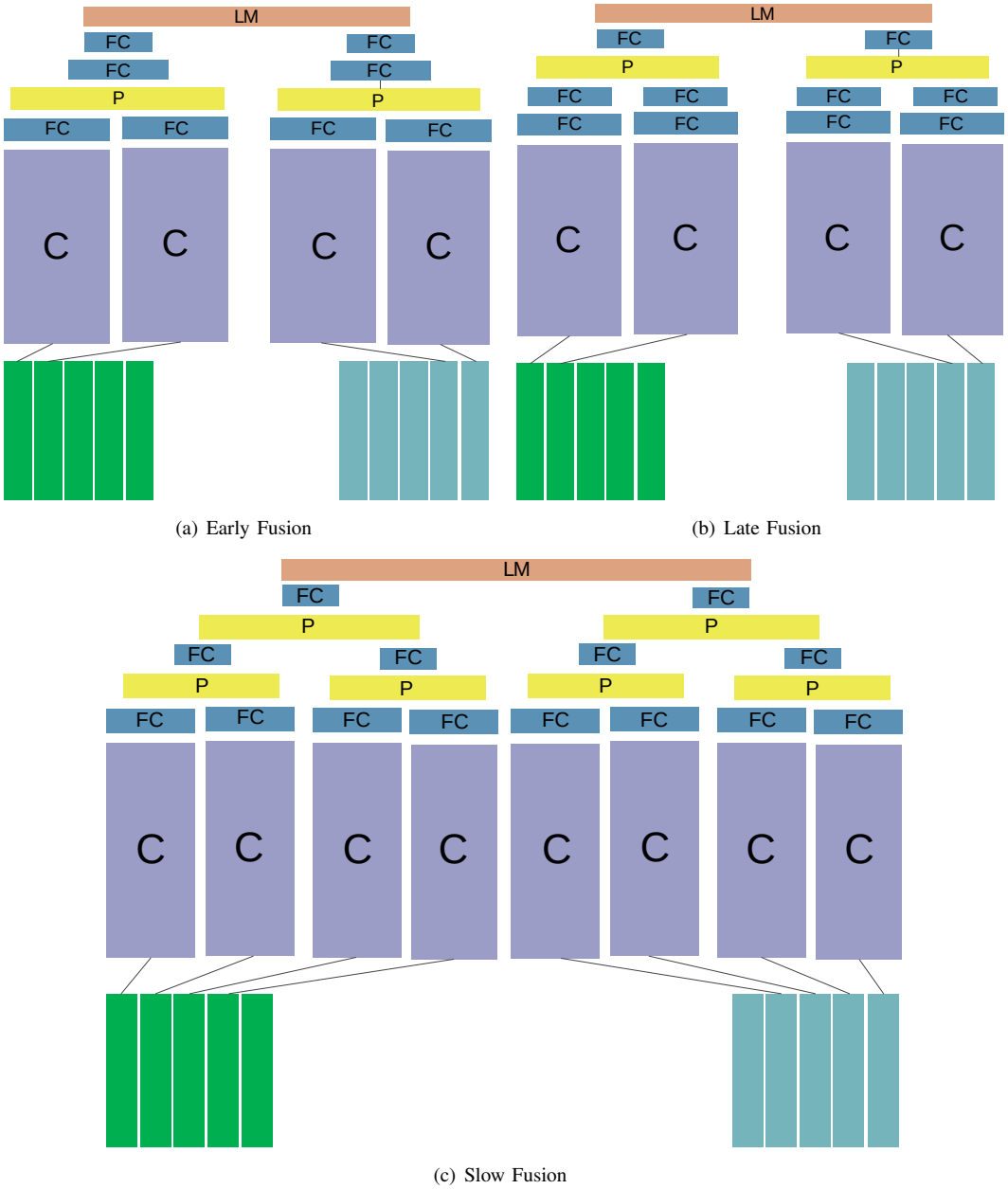


Fig. 2. Different DVH architectures with frame fusion, where the vertical bars represent the raw image frames and each color represents a single video, 'C' represents the stacked convolution and pooling layers, 'FC' represents the fully-connected layers, 'P' represents the temporal pooling layers, and 'LM' represents the large-margin cost function, respectively.

fully connected layers at the end. By letting $s(\cdot)$ be the output at the last fully connected layer where it contains K nodes, the binary code for the set of image frames of the ith video is computed as follows:

$$\mathbf{b}_u = \text{sgn}(s(\mathbf{I}_u)) \tag{4}$$

There are two important intuitions for our DVH model: (1) By performing several nonlinear transformations with a discriminative criterion, more robust visual representation can be obtained. While kernel-based models can provide explicit nonlinear mappings, pre-defined kernel functions cannot well capture the nonlinearity of samples; (2) By performing the temporal pooling, we can implicitly exploit the relevant frames and extract a balance of global and local information from video frames. By doing so, the noisy frames which may degrade the quality of the binary codes can be implicitly ignored. We discuss how to exploit both the temporal and discriminative information in our deep architecture as follows:

1) Temporal Information: Since videos typically represent motion-based features across time, it is necessary to consider temporal information to fully obtain video-level representation. In order to exploit temporal information in deep networks, we perform pooling operations across frames between the fully-connected layers. In our work, we perform fusion of temporal information through average

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2016.2645404, IEEE Transactions on Multimedia

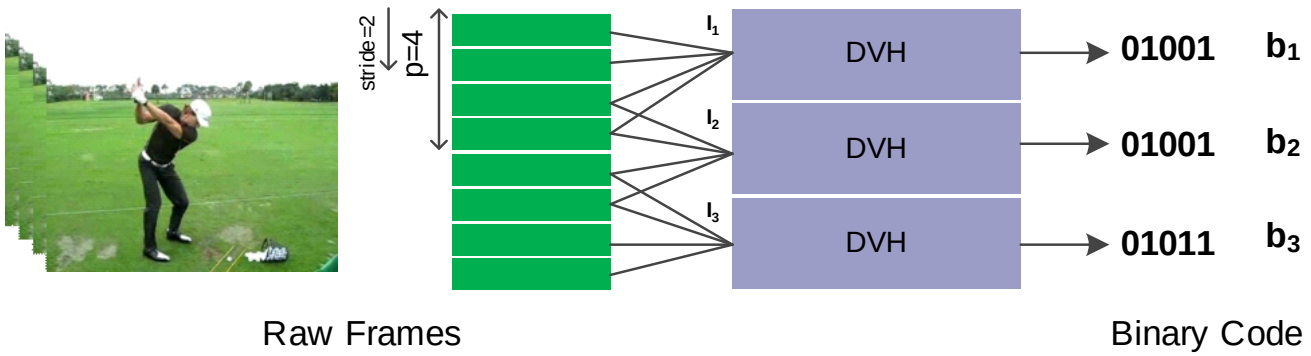


Fig. 3. Extracting binary codes for a single video from DVH. We extract the binary code in a pre-defined number of consecutive frames, p, and shift to the next set based on a fixed stride value. In this figure, given 8 frames, the DVH fuses $p = 4$ frames with a stride of 2, resulting to 3 final compact binary codes.

pooling[2]. We describe three deep networks with various feature pooling architectures:

Early Fusion: The early fusion architecture first passes image frames through the convolution and pooling layers and then fuses the information at the first fully connected layer immediately, as shown in Fig. 2(a).

Late Fusion: The late fusion architecture first passes image frames through the convolution and pooling layers up to the other fully-connected layers and then fuses the information at the last fully connected layer, as shown in Fig. 2(b).

Slow Fusion: The slow fusion architecture is a balance of the early and late fusion. Image frames are passed through the convolution and pooling layers and then fused in a hierarchical manner such that smaller temporal windows are used as it approaches the top layer. In this work, a two-stage fusion strategy is implemented. Fig. 2(c) details the architecture of this fusion strategy.

2) Discriminative and Binary Information: To learn the parameters in the deep network discriminatively, we employ the Siamese network [5] with a large-margin learning framework rather than the conventional contrastive divergence criterion [18]. Specifically, we present a new formulation which consists of two new objective criterions for binary code learning. The first objective performs discriminative learning. Specifically, given two sets of image frames, $\mathbf{I}_u$ and $\mathbf{I}_v$, we minimize the intra-class variation and maximize the inter-class variation of the binary feature representation at the top layer of these two networks, simultaneously. Given their Hamming distance $d_{u,v}(\mathbf{b}_u, \mathbf{b}_v)$ at the top layer, where $\mathbf{b}_u = \text{sign}(s(\mathbf{I}_u))$ and $\mathbf{b}_v = \text{sign}(s(\mathbf{I}_v))$, we expect that $d_{u,v}$ is small if u and v are the same class, and large if they are from different classes, which is formulated as the following constraints:

$$d_{u,v}(\mathbf{b}_u, \mathbf{b}_v) \leq \theta_1, \quad \text{if } y_u = y_v \tag{5}$$

$$d_{u,v}(\mathbf{b}_u, \mathbf{b}_v) \geq \theta_2, \quad \text{if } y_u \neq y_v \tag{6}$$

where θ_1 and θ_2 are the small and large thresholds, respectively.

[2]We found that max pooling does not give very representative information compared to average pooling.

By combining (5) and (6), we have the following formulas:

$$\delta_{u,v} \left(\theta - d_{u,v}(\mathbf{b}_u, \mathbf{b}_v)\right) > 1 \tag{7}$$

where $\theta_1 = \theta - 1$ and $\theta_2 = \theta + 1$, and $\delta_{u,v} = 1$ means that u and v are from the same class, and $\delta_{u,v} = -1$ indicates that they are from different classes.

This leads to the following objective function:

$$J_1 = f(1 - \delta_{u,v}(\theta - d_{u,v}(\mathbf{b}_u, \mathbf{b}_v))) \tag{8}$$

where $f(z)$ is a generalized logistic loss function which is a smooth approximation of the hinge loss function: $z = \max(z, 0)$, and defined as follows:

$$f(z) = \frac{1}{\rho} \log(1 + \exp(\rho z)) \tag{9}$$

where ρ is the sharpness parameter set to 10 and θ is the threshold parameter set to $K/4$.

The second objective is to ensure efficient binary codes by minimizing the quantization loss [16] between real-valued codes and binary codes as follows:

$$J_2 = \|s(\mathbf{I}_u) - \mathbf{b}_u\|_F^2 + \|s(\mathbf{I}_v) - \mathbf{b}_v\|_F^2 \tag{10}$$

Hence, the final objective function for our DVH is formulated as:

$$\begin{aligned} \min_{\mathbf{b}_v, \mathbf{b}_v} J &= J_1 + \lambda J_2 \\ &= f(1 - \delta_{u,v}(\theta - d_{u,v}(\mathbf{b}_u, \mathbf{b}_v))) \\ &+ \lambda(\|s(\mathbf{I}_u) - \mathbf{b}_u\|_F^2 + \|s(\mathbf{I}_v) - \mathbf{b}_v\|_F^2) \end{aligned} \tag{11}$$

where J_1 exploits the discriminative information, J_2 minimizes the quantization loss, and λ is a constant parameter which balances the two criterions.

We use the standard mini-batch gradient descent and back-propagation to solve the optimization problem. We first relax the binary constraints in the first objective and use the signed magnitude real codes during back-propagation. Given $\mathbf{h}_u = s(\mathbf{I}_u)$ and $\mathbf{h}_v = s(\mathbf{I}_v)$ are the real-value code values from the top layer, the back-propagation is implemented first by taking the derivative of J with respect

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2016.2645404, IEEE Transactions on Multimedia

to $\mathbf{h}_u$ and $\mathbf{h}_v$:

$$\frac{\partial J}{\mathbf{h}_u} = f'(z)\delta_{u,v}(\mathbf{h}_u - \mathbf{h}_v) + \lambda(\mathbf{h}_u - \mathbf{b}_u) \quad (12)$$

$$\frac{\partial J}{\mathbf{h}_v} = f'(z)\delta_{u,v}(\mathbf{h}_v - \mathbf{h}_u) + \lambda(\mathbf{h}_v - \mathbf{b}_v) \quad (13)$$

These gradients are then back-propagated to update the weights at the earlier layers.

C. Extracting Binary Codes from One Video

Since the number of frames are usually different for different videos, we extract multiple binary codes for each video by using our DVH approach through a set of consecutive image frames in the video with a specific stride. Fig. 3 shows the hashing procedure for one video by using our model. The over-all hamming distance, D_H, for a pair of videos are then obtained by getting the average of the hamming distances, d_H, for each pair of binary codes as follows:

$$D_H(\mathbf{X}_i, \mathbf{X}_j) = \frac{1}{g_i g_j} \sum_{u=1}^{g_i} \sum_{v=1}^{g_j} d_H(\mathbf{b}_u, \mathbf{b}_v) \quad (14)$$

where g_i and g_j is the number of frame sets for videos $\mathbf{X}_i$ and $\mathbf{X}_j$. By doing so, we are able to compare the similarity of videos with less computation than using frame to frame comparisons, which are more representative than using single features for each video.

D. Implementation Details

Our deep network composes of a stacked convolutional and pooling layers with parameters obtained from pre-trained models, and connected to a series of fully connected layers such that the top-most layer contains K-dimensional features. The hidden fully connected layers use rectified linear unit (ReLU) as the activation function, while the top-most layer has a hyperbolic tangent activation to ensure centered feature and have balanced {-1,1} values. The parameters in the fully connected layers are initialized using the Xavier initialization [15][3]. To be consistent, all models have a fully-connected layers of dimensions $[4096 \rightarrow 500 \rightarrow 200 \rightarrow K]$. To avoid over-fitting, we enable training only in the fully connected layers. Our deep architecture and experiments were implemented under the MatConvNet [54] framework. The learning rate, momentum, and weight decay were set to 0.002, 0.9, and 0.0001, respectively. Table I summarizes the implementations of different DVH methods after the stacked convolutional and pooling layers. At the training stage, we iteratively passed through all the training videos where we randomly chose video pairs. For each video pair, we randomly chose a set of p successive frames and then packed them into batches to pass into the network. We ensure that the positive and negative pairs for each batch are in an approximate 1:2 ratio. The training procedure converged when the loss does not change within a certain threshold for an epoch. For all experiments, ρ was set to 10 based on the empirical testing to obtain a smooth approximation for the Hinge Loss. We experimented with different values, and found that the results appear to be particularly insensitive to ρ.

[3]We initialize the biases to be 0 and the weights at each layer as $\mathbf{W} = U\left[-\sqrt{\frac{6}{n_{in}+n_{out}}}, \sqrt{\frac{6}{n_{in}+n_{out}}}\right]$ where $\mathbf{W} \in \mathbb{R}^{n_{in} \times n_{out}}$.

TABLE I
DVH IMPLEMENTATIONS IN THE EXPERIMENTS.

Early Fusion	Late Fusion	Slow Fusion
fc7 - 4096	fc7 - 4096	fc7 - 4096
pool at p frames	-	pool at p_1 frames
fc8 - 500	fc8 - 500	fc8 - 500
-	pool at p frames	pool at p_2 frames
fc9 - 200	fc9 - 200	fc9 - 200
K	K	K

IV. Experiments

To evaluate the effectiveness of our proposed DVH method for scalable video search, we conducted experiments on two video datasets namely the CCV and JHMDB datasets. Figs. 4-5 show some sample video frames from these two datasets. The details of the experiments and the results are described in the following sections.

A. Datasets and Experimental Settings

Columbia Consumer Video (CCV) dataset [28]: It consists of 9,317 videos with an average duration of 80 seconds extracted from YouTube. The videos were categorized to 20 different categories such as *basketball*, *wedding* and *music performance*. Similar to [61], we sampled frames every 2 seconds and ensured that each video had a minimum of 30 frames. Since most of the categories in this dataset are events, the videos contain large variations among frames making the task very challenging. In our experiments, we randomly selected 20 videos per category for training, 25 videos per category as the query data, and 100 videos per category as the gallery data. This results in 400, 500, and 2000 videos for the training, query, and gallery sets, respectively.

For our deep model, we used the pre-trained VGG-net [51] as our stacked convolution and pooling layers. We used a batch size of 200 and a frame size of $p = 10$. For our Slow Fusion architecture, the first pooling layer fuses the data of $p_1 = 5$ frames with a stride of 2, the second pooling layer fuses the data of the final $p_2 = 3$ frames. For testing samples, we obtained the binary codes for each video at a frame stride of 5.

Joint-annotated HMDB (JHMDB) dataset [24]: It consists of 928 action videos having 36 to 55 frames per video, which was taken from the HMDB dataset [32] for human motion recognition. The action videos are categorized into 21 human actions such as *brushing hair*, *clapping*, and *climbing*. Although action recognition makes use of flow and RGB information [4], we only used the full body optical flow representation for simplicity. In our experiments, we randomly selected 10 videos per category as training samples, 10 videos per category as query samples, and 20

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2016.2645404, IEEE Transactions on Multimedia

Fig. 4. Two sample videos for the Columbia Consumer Video (CCV) database. The first video belongs to the *music performance* category, while the second video belongs to the *baseball* category.

Fig. 5. Two sample videos for the Joint-annotated HMDB (JHMDB) database. The first video represents a *shooting* action video, while the second video represents a *catching* action video.

videos per category as gallery samples. This results in 210, 210, and 420 videos for the training, query, and gallery sets, respectively.

For our deep model, we used the CNN motion network by [14] as our pre-trained stacked convolution and pooling layers. We used a batch size of 50 and a frame size of $p = 10$. For our Slow Fusion architecture, the first pooling layer fuses the data of $p_1 = 5$ frames with a stride of 2, the second pooling layer fuses the data of the final $p_2 = 3$ frames. For testing samples, we obtained the binary codes at a stride of 2.

B. Evaluation Metrics

To measure the performance of our DVH, we used the Hamming ranking and Hamming look-up as evaluation metrics to compare the performance of different methods. For Hamming ranking, the mean Average Precision (mAP) and Precision@N are evaluated. The mAP is defined as the mean of the average precision of the top retrieved samples across all queries, while Precision@N is defined as the percentage of true labels among the top N retrieved samples. For Hamming look-up, the precision when the hamming radius is set as $r = 2$ is evaluated where it measures the precision over all the samples that is within a hamming radius of $r = 2$. At $K = 64$, Hamming look-up precision is not evaluated because it will be impractical for longer bit lengths.

C. Experimental Results

Comparison with Different Deep Baselines: We first compared our DVH with three baseline deep architectures which do not use temporal fusion in the fully-connected layers. The baseline methods are described as follows:

Single-Frame: In the single-frame model, we considered each frame of the video as a single image with its own label information. Similar to DVH, we used the large-margin criterion for the Siamese network to learn the parameters. However, we only used single frames as the input and do not perform temporal fusion. The cost function is defined as:

$$J = f(1 - \delta_{u,v}(\theta - d_{u,v}(s(\mathbf{x}_u), s(\mathbf{x}_v)))) \tag{15}$$

Single-Frame + Temporal: In the single-fame + temporal model, we exploited the temporal information with the same large-margin criterion so that the frames which are close to each other are similar as much as possible, defined as below:

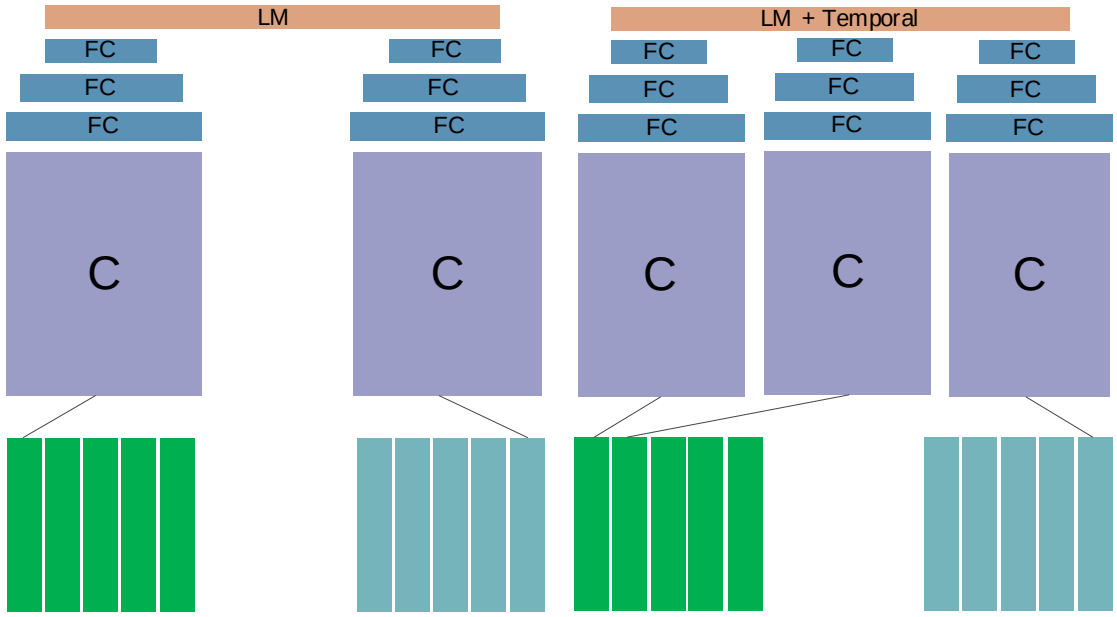
$$\begin{aligned} J &= f(1 - \delta_{u_1,v}(\theta - d_{u_1,v}(s(\mathbf{x}_{u_1}), s(\mathbf{x}_v)))) \\ &+ \nu \|s(\mathbf{x}_{u_1}) - s(\mathbf{x}_{u_2})\|_2^2 \end{aligned} \tag{16}$$

where ν is the balancing term, and u_1 and u_2 are two randomly selected image frames from the same video which are apart by a minimum of five frames. In the experiments, we used $\nu = 0.1$. Fig. 6 shows the architectures of the first two baseline methods.

Video-Level Feature: In this model, we pooled all frame-level features from the single-frame deep model to compute video-level features to evaluate the large margin criterion. By doing so, we obtained a representative global binary vector for each video.

Tables II and III show the performance of different methods on the CCV and JHMDB datasets, respectively. As can be seen, our DVH architectures outperform the other deep

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2016.2645404, IEEE Transactions on Multimedia



(a) Single Frame (b) Single Frame + Temporal

Fig. 6. Deep baseline architectures based on frame-by-frame training for video hashing. The first baseline is similar to image hashing where each frame is a single image. The second baseline adds a temporal criterion during training such that given two frames that are temporally close should have a similar compact feature as much as possible.

TABLE II
RESULTS ON THE CCV DATASET IN COMPARISON WITH THE BASELINE DEEP ARCHITECTURE.

Method	Hamming ranking (mAP, %)			precision (%) @ N = 100			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
Single	32.62	34.23	35.02	37.37	38.86	38.50	23.38	5.63
Single+Temporal	33.60	35.60	37.27	38.05	39.74	40.93	30.76	16.44
Video-Level	29.45	30.79	29.19	34.44	35.98	34.05	22.48	11.11
Early Fusion	37.18	40.86	**41.54**	40.11	41.89	**42.41**	36.61	22.80
Late Fusion	**38.54**	**41.08**	41.51	**40.29**	**42.08**	42.23	**37.32**	**23.10**
Slow Fusion	38.27	40.80	41.41	39.95	41.88	42.34	36.55	23.06

TABLE III
RESULTS ON THE JHMDB DATASET IN COMPARISON WITH THE BASELINE DEEP ARCHITECTURE.

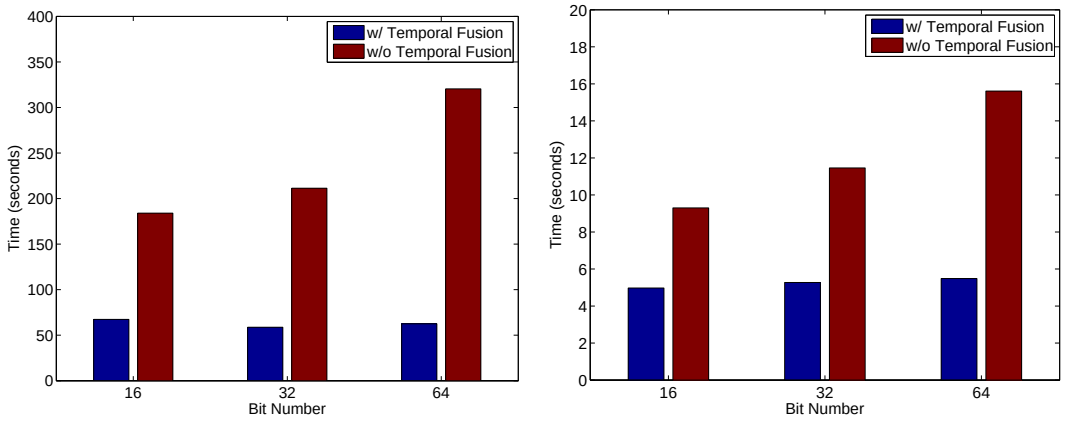
Method	Hamming ranking (mAP, %)			precision (%) @ N = 10			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
Single	32.73	33.89	31.74	42.67	43.81	44.05	6.19	0
Single+Temporal	33.19	34.85	35.58	43.81	45.33	45.19	9.05	0
Video-Level	30.07	30.95	32.53	39.10	41.67	41.86	6.58	0
Early Fusion	**35.19**	**37.43**	**37.95**	**46.48**	47.62	48.19	**31.31**	**12.46**
Late Fusion	34.93	36.78	37.53	45.10	**48.05**	48.14	29.67	10.10
Slow Fusion	34.86	36.59	36.63	44.52	47.90	**48.24**	25.25	8.67

learning based baseline architectures. It is interesting to see that the second baseline (Single+Temporal) beats the first baseline (Single), which shows that temporal information is important. The video-level features also yielded competitive representations but often achieved worse performance than the single-frame deep model. The late fusion and early fusion DVH architectures obtain the best performance on the CCV and JHMDB datasets, respectively. For CCV, a late fusion would mean it prefers to exploit high-level global information, while for JHMDB an Early Fusion shows it prefers to exploit local motion information. This is reasonable because JHMDB focuses on action information while CCV is more on events which are generally holistic.

We also examined the retrieval time of different deep baseline architectures, which is shown in Fig. 7. As can be seen, a longer retrieval time is necessary since the deep baseline architectures transform each image frame into a binary code. Differently, our DVH method performs temporal fusion so fewer binary code comparisons are implemented resulting in a faster retrieval time for the whole query-gallery set. This is most obvious on the CCV dataset since more frames are present in a single video, and larger gallery and query videos are used.

Comparison with State-of-the-Art Learning-based

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2016.2645404, IEEE Transactions on Multimedia



(a) CCV (b) JHMDB

Fig. 7. Retrieval time on the (a) CCV and (b) JHMDB datasets, respectively.

TABLE IV
RESULTS OF DIFFERENT LEARNING-BASED HASHING METHODS ON THE CCV DATASET.

Method	Hamming ranking (mAP, %)			precision (%) @ N = 100			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
PCAH [55]	20.83	21.45	19.37	25.80	26.50	25.51	3.03	0
PCA-ITQ [16]	22.49	24.13	24.42	27.71	28.99	29.61	13.43	0
AGH [40]	14.91	15.22	11.24	20.52	23.37	20.16	13.43	1.58
KSH [39]	32.43	34.34	35.40	36.27	38.33	38.75	18.27	7.64
CCA-ITQ [16]	36.58	38.18	38.32	39.13	40.41	40.51	16.15	7.17
FastHash [36]	34.72	38.37	38.47	38.83	40.85	41.37	12.73	5.36
DVH	**38.54**	**41.08**	**41.51**	**40.29**	**42.08**	**42.23**	**37.32**	**23.10**

TABLE V
RESULTS OF DIFFERENT LEARNING-BASED HASHING METHODS ON THE JHDMB DATASET.

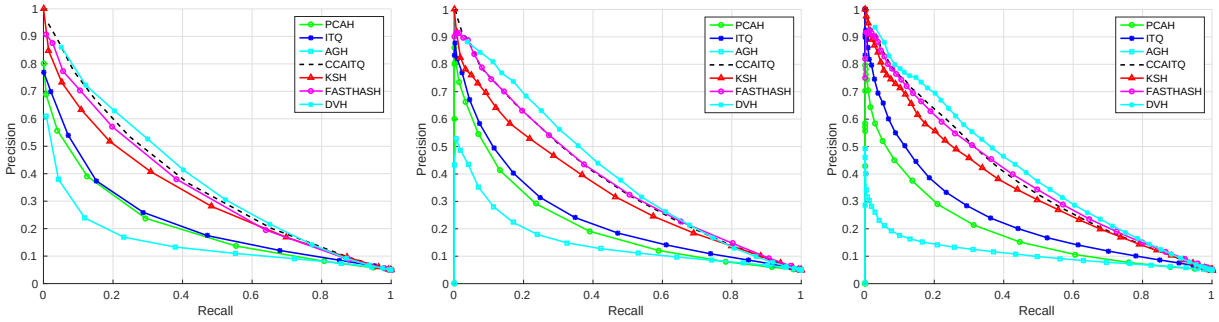
Method	Hamming ranking (mAP, %)			precision (%) @ N = 10			precision (%) @ r=2	
	16	32	64	16	32	64	16	32
PCAH [55]	16.89	17.64	18.30	27.05	31.31	31.81	0	0
PCA-ITQ [16]	13.80	14.16	14.44	22.52	25.19	27.05	0.58	0
AGH [40]	13.74	14.30	16.90	23.38	26.86	27.57	0.12	0
KSH [39]	27.50	28.32	33.51	39.14	39.05	42.38	0	0
CCA-ITQ [16]	27.20	31.96	31.44	43.48	47.56	47.43	1.51	0
FastHash [36]	31.19	33.72	36.63	42.29	44.33	46.52	0	0
DVH	**35.19**	**37.43**	**37.95**	**46.48**	**47.62**	**48.42**	**31.31**	**12.46**

Hashing methods: We also compared our DVH method with several popular hashing methods including PCA Hashing [55], PCA-ITQ [16], Anchor Graph Hashing (AGH) [40], Kernel Supervised Hashing (KSH) [39], CCA-ITQ [16], and FastHash [36]. Specifically, PCAH, PCA-ITQ and AGH are unsupervised hashing methods, and KSH, CCA-ITQ, and FastHash exploit the label information of samples to learn discriminative hash codes. The standard implementations of all methods are from the original authors and the default parameters were set based on their respective papers. For consistency, the experiments were carried out with the same selected training, gallery and query sets. For the different hashing methods being compared, we considered each frame as an image and encoded its respective binary code based on the 4096-dimension CNN feature obtained from the fully-connected layer of the pre-trained models used, and defined the hamming distance of two videos as the average of all hamming distances between images from each video.

Tables IV and V show the performance of different hashing methods on the CCV and JHMDB experiments, respectively. We found that the DVH architecture yielded the best performance, where the Late fusion was for the CCV dataset, and the Early fusion was for the JHMDB dataset, respectively. As can be seen, our method consistently outperforms the other existing hashing methods. Most surprising is the hamming look-up precision (HLP) evaluation results which show significant improvement across varying bit lengths. This shows that representing the video in a deep nonlinear binary feature vector gives strong representation for retrieval. Figs. 8-11 show the recall and precision curve, and precision curves vs the retrieval number N on the CCV and JHMDB datasets. We see that our method outperforms the compared methods in most scenarios.

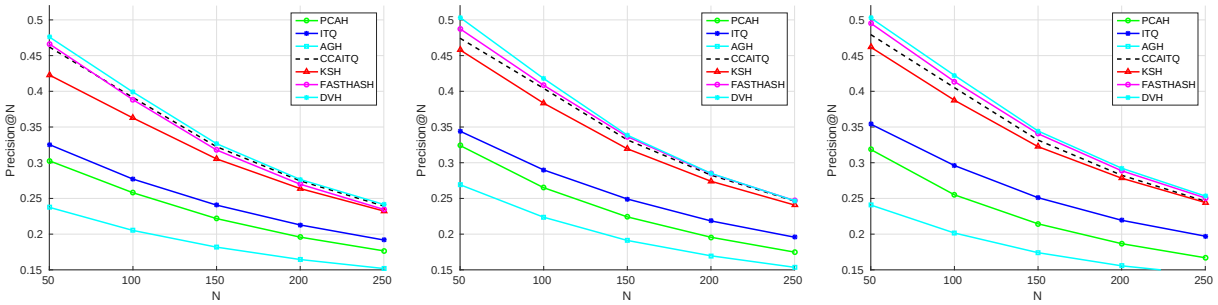
Comparison with Different Video Hashing Meth-

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2016.2645404, IEEE Transactions on Multimedia



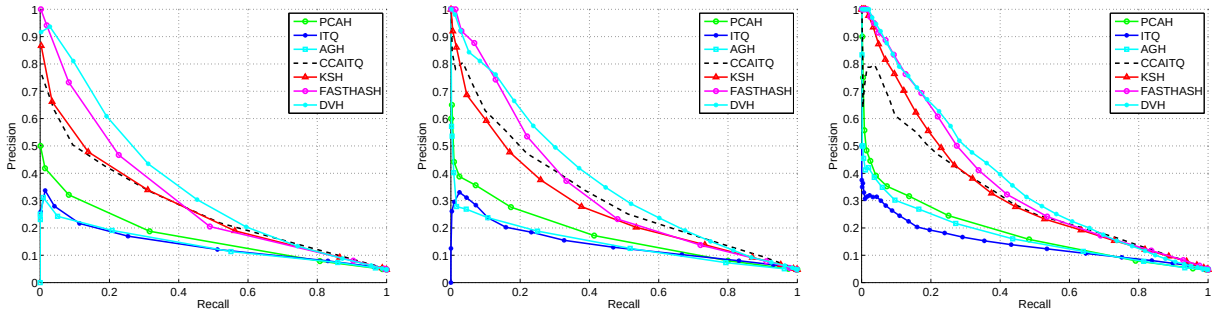
(a) PR curve (16 bits) (b) PR curve (32 bits) (c) PR curve (64 bits)

Fig. 8. Precision-Recall (PR) curves on the CCV dataset versus varying code lengths.



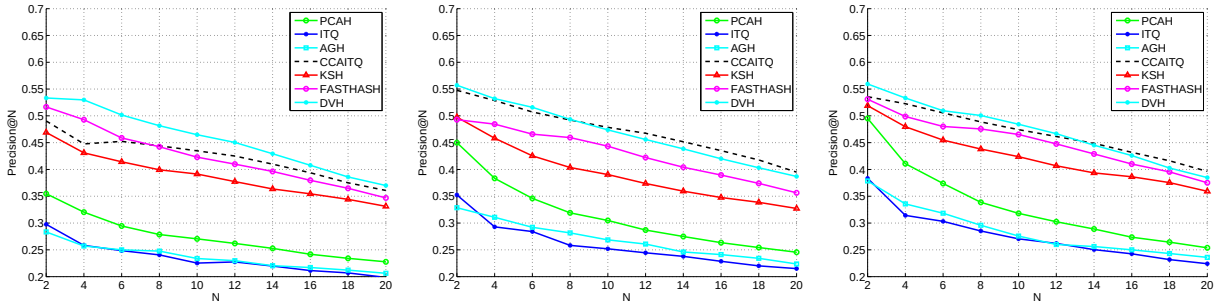
(a) Prec-N curve (16 bits) (b) Prec-N curve (32 bits) (c) Prec-N curve (64 bits)

Fig. 9. Precision-N curves on the CCV dataset versus varying code lengths.



(a) PR curve (16 bits) (b) PR curve (32 bits) (c) PR curve (64 bits)

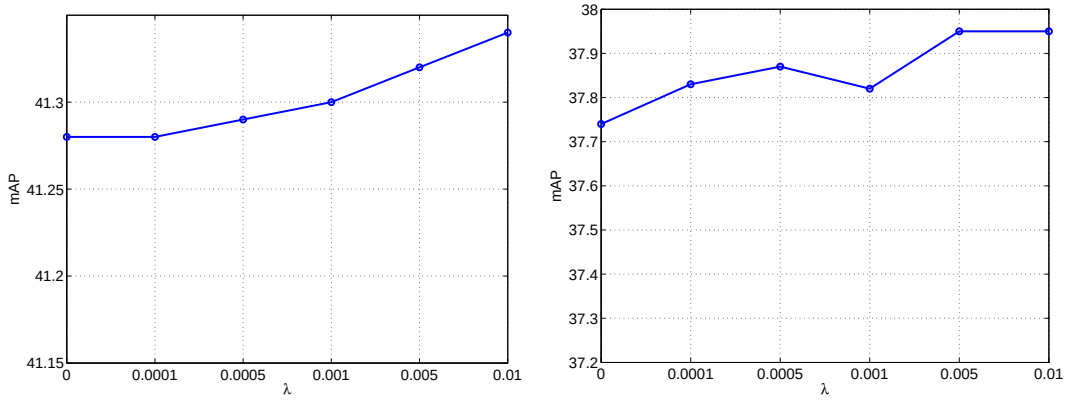
Fig. 10. Precision-Recall (PR) curves on the JHMDB dataset versus varying code lengths.



(a) Prec-N curve (16 bits) (b) Prec-N curve (32 bits) (c) Prec-N curve (64 bits)

Fig. 11. Precision-N curves on the JHMDB dataset versus varying code lengths.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TMM.2016.2645404, IEEE Transactions on Multimedia



(a) CCV (b) JHMDB

Fig. 12. The mAP performance of our DVH method at varying λ for the 64-bit experiment on the (a) CCV and (b) JHMDB datasets, respectively.

TABLE VI
HAMMING RANKING (MAP, %) RESULTS ON THE CCV DATASET IN COMPARISON WITH OTHER VIDEO HASHING ALGORITHMS.

Method	16 bits	32 bits	64 bits
DVH-CNN	**38.54**	**41.08**	**41.51**
ITQ-CNN	22.49	24.13	24.42
ITQ-SIFT	12.33	14.63	14.52
VHDT-SIFT [61]	11.40	13.00	15.90
CVC-CNN [34]	27.53	32.16	36.14

TABLE VII
HAMMING RANKING (MAP, %) RESULTS OF OUR DVH ON THE CCV DATASET IN DIFFERENT VALUES OF p AND s.

Method	16 bits	32 bits	64 bits
$p = 2, s = 2$	37.81	39.48	40.46
$p = 5, s = 5$	37.81	40.78	41.09
$p = 10, s = 5$	**38.54**	**41.08**	**41.51**
$p = 20, s = 5$	37.25	40.11	41.31

TABLE VIII
HAMMING RANKING (MAP, %) RESULTS OF OUR DVH ON THE JHMDB DATASET IN DIFFERENT VALUES OF p AND s.

Method	16 bits	32 bits	64 bits
$p = 2, s = 2$	32.48	32.58	33.55
$p = 5, s = 5$	33.21	35.32	35.30
$p = 10, s = 5$	**35.19**	**37.46**	**37.95**
$p = 20, s = 5$	37.97	35.82	36.43

ods: We compared our method with two video hashing methods as shown in Table VI. For Video Hashing with both Discriminative commonality and Temporal consistency (VHDT) [61], the mAP results were obtained from the original paper. However, their method used SIFT BoW features so it is difficult to directly compare. To approximate, we have applied ITQ-SIFT and found that it is comparable with VHDT. Our DVH is much better than ITQ-CNN. This is because VHDT performs linear transformations which may not really capture the nonlinearity of data in videos.

For the Compact Video Coding (CVC) method [34], we used the publicly released code, tuned the parameters to obtain the best possible result and used CNN features to construct the covariance feature for each video. As can be seen, our DVH outperforms CVC at all bit lengths. This is because CVC converted the whole video into a single feature code which may lead to loss of temporal information, while our DVH method considered the temporal and discriminative information of each video.

Parameter Analysis: We also analyzed the varying values of λ during training to see the contribution of the two criterions in the over-all performance of our DVH method. Fig. 12 shows the mAP performance of our DVH method at $\lambda = [0, 0.01, 0.005, 0.001, 0.0005, 0.0001]$ for the CCV and JHMDB datasets. As expected, we see that the discriminative information makes most of the contribution since even at $\lambda = 0$, the performance is very competitive. Nevertheless, minimizing the quantization loss still provides improvement in the over-all performance. However, it is important to see that the quantization loss criterion should not overpower the discriminative criterion. In our experiments, the optimal value for λ is at a range of $[0.005, 0.01]$.

We also conducted experiments on varying values of the frame size p, which is the number of frames as the input in the deep network to obtain a binary code, and the stride s, which is the number of non-overlapped frames. We used the best fusion algorithm for each dataset (CCV-Late, JHMDB-Early). As can be seen in Table VII and VIII, our method shows a decline in mAP at $p < 10$ probably because frames are very much similar which do not really exploit the temporal information. Similarly, a much higher p may also reduce the performance because it extracts more global video features.

D. Discussion

The above experimental results suggest the following three key observations:

1) Our deep video hashing method achieves very competitive performance compared to other deep baseline architectures which shows that performing temporal fusion during training contributes well to the over-all performance. In addition, retrieval time is also

reduced because of the temporal fusion.

2) Our DVH outperforms state-of-the-art image-based hashing methods which shows that the binary codes obtained from our hashing method are strong representations due to the discriminative training we employed. Furthermore, our DVH also outperforms other video hashing methods by a large margin.
3) The large-margin criterion yields the largest contribution in our DVH method. However, the binary quantization term also provides improvements in the over-all performance. For the parameter p, we see that the best performance can be obtained when the parameter of p is set to 10 because it is a good balance of extracting global and local video features.

V. Conclusion

In this paper, we have proposed a deep video hashing approach with various frame pooling architectures to learn binary codes for each video in a deep framework such that both temporal and discriminative information are well exploited. Experimental results on two video databases clearly demonstrate that our method achieved better performance with the state-of-the-art hashing methods.

There are two interesting directions for future work:

1) Our DVH method composed of frame-level pooling layers to exploit temporal information. It is interesting to incorporate more complex networks such as recurrent neural networks (RNN) [47], long short term memory (LSTM) [19] and 3D-CNNs [26] to further improve the performance.
2) In this work, we learned our DVH network using supervised information. Hence, it is interesting to learn a deep network using quantization-based [16], [23] criterions, which does not exploit label information.

Acknowledgement

This work is supported by the National Key Research and Development Program of China under Grant 2016YFB1001001, the National Natural Science Foundation of China under Grants 61672306, and the National 1000 Young Talents Plan Program. Partial of this work was carried out at the Rapid-Rich Object Search (ROSE) Lab at the Nanyang Technological University Singapore.

References

[1] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *Human Behavior Understanding*, pages 29–39. 2011.
[2] L. Cao, Z. Li, Y. Mu, and S.-F. Chang. Submodular video hashing: a unified framework towards video pooling and indexing. In *ACM MM*, pages 299–308, 2012.
[3] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen. Deep quantization network for efficient image retrieval. In *AAAI*, pages 3457–3463, 2016.
[4] G. Chéron, I. Laptev, and C. Schmid. P-cnn: pose-based cnn features for action recognition. In *ICCV*, pages 3218–3226, 2015.
[5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, pages 539–546, 2005.
[6] C. L. Chou, H. T. Chen, and S. Y. Lee. Pattern-based near-duplicate video retrieval and localization on web-scale videos. *TMM*, 17(3):382–395, 2015.
[7] B. Coskun, B. Sankur, and N. Memon. Spatio–temporal transform based video hashing. *TMM*, 8(6):1190–1208, 2006.
[8] M. Douze, H. Jégou, and C. Schmid. An image-based approach to video copy detection with spatio-temporal post-filtering. *TMM*, 12(4):257–266, 2010.
[9] L.-Y. Duan, J. Lin, Z. Wang, T. Huang, and W. Gao. Weighted component hashing of binary aggregated descriptors for fast visual search. *TMM*, 17(6):828–842, 2015.
[10] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *CVPR*, pages 2475–2483, 2015.
[11] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *TPAMI*, 35(8):1915–1929, 2013.
[12] A. Gionis, P. Indyk, R. Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.
[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, pages 580–587, 2014.
[14] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, pages 759–768, 2015.
[15] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *ICAIS*, pages 249–256, 2010.
[16] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824, 2011.
[17] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, pages 2938–2945, 2013.
[18] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.
[19] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
[20] S. C. Hoi and M. R. Lyu. A multimodal and multilevel ranking scheme for large-scale video retrieval. *TMM*, 10(4):607–619, 2008.
[21] W. Hu, N. Xie, L. Li, X. Zeng, and S. Maybank. A survey on visual content-based video indexing and retrieval. *TSCVT*, 41(6):797–819, 2011.
[22] G. Irie, Z. Li, X.-M. Wu, and S.-F. Chang. Locally linear hashing for extracting non-linear manifolds. In *CVPR*, pages 2115–2122, 2014.
[23] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *TPAMI*, 33(1):117–128, 2011.
[24] H. Jhuang, J. Gall, S. Zuffi, C. Schmid, and M. Black. Towards understanding action recognition. In *ICCV*, pages 3192–3199, 2013.
[25] R. Ji, L.-Y. Duan, J. Chen, L. Xie, H. Yao, and W. Gao. Learning to distribute vocabulary indexing for scalable visual search. *TMM*, 15(1):153–166, 2013.
[26] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *TPAMI*, 35(1):221–231, 2013.
[27] Y.-G. Jiang, J. Wang, X. Xue, and S.-F. Chang. Query-adaptive image search with hash codes. *TMM*, 15(2):442–453, 2013.
[28] Y.-G. Jiang, G. Ye, S.-F. Chang, D. Ellis, and A. C. Loui. Consumer video understanding: A benchmark database and an evaluation of human and machine performance. In *ACM MM*, pages 29–37, 2011.
[29] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732, 2014.
[30] C. Kofler, M. Larson, and A. Hanjalic. Intent-aware video search result optimization. *TMM*, 16(5):1421–1433, 2014.
[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.
[32] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre. Hmdb: a large video database for human motion recognition. In *ICCV*, pages 2556–2563, 2011.
[33] P. Li, M. Wang, J. Cheng, C. Xu, and H. Lu. Spectral hashing with semantically consistent graph for image indexing. *TMM*, 15(1):141–152, 2013.
[34] Y. Li, R. Wang, Z. Cui, S. Shan, and X. Chen. Compact video code and its application to robust face retrieval in tv-series. In *BMVC*, pages 1–12, 2014.
[35] Y. Li, R. Wang, S. Shan, and X. Chen. Hierarchical hybrid statistic based video binary code and its application to face retrieval in tv-series. In *FG*, pages 1–8, 2015.
[36] G. Lin, C. Shen, Q. Shi, A. Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, pages 1963–1970, 2014.
[37] Z. Lin, G. Ding, M. Hu, and J. Wang. Semantics-preserving hashing

for cross-view retrieval. In *CVPR*, pages 3864–3872, 2015.
[38] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *NIPS*, pages 3419–3427, 2014.
[39] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.
[40] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.
[41] X. Liu, X. Fan, C. Deng, Z. Li, H. Su, and D. Tao. Multilinear hyperplane hashing. In *CVPR*, pages 5119–5127, 2016.
[42] X. Liu, L. Huang, C. Deng, B. Lang, and D. Tao. Query-adaptive hash code ranking for large-scale multi-view visual search. *TIP*, 25(10):4514–4524, 2016.
[43] X. Liu, Y. Mu, D. Zhang, B. Lang, and X. Li. Large-scale unsupervised hashing with shared structure learning. *TSCVT*, 45(9):1811–1822, 2015.
[44] J. Masci, M. M. Bronstein, A. M. Bronstein, and J. Schmidhuber. Multimodal similarity-preserving hashing. *TPAMI*, 36(4):824–830, 2014.
[45] Y. Mu, G. Hua, W. Fan, and S.-F. Chang. Hash-svm: Scalable kernel machines for large-scale visual classification. In *CVPR*, pages 979–986, 2014.
[46] X. Peng and C. Schmid. Multi-region two-stream r-cnn for action detection. In *ECCV*, pages 744–759, 2016.
[47] D. Rumelhart, G. Hinton, and R. Williams. Learning sequential structure in simple recurrent networks. *Parallel distributed processing: Experiments in the microstructure of cognition*, 1986.
[48] J. Shao, C.-C. Loy, K. Kang, and X. Wang. Slicing convolutional neural network for crowd video understanding. In *CVPR*, pages 5620–5628, 2016.
[49] F. Shen, C. Shen, Q. Shi, A. Van Den Hengel, and Z. Tang. Inductive hashing on manifolds. In *CVPR*, pages 1562–1569, 2013.
[50] F. Shen, C. Shen, Q. Shi, A. van den Hengel, Z. Tang, and H. T. Shen. Hashing on nonlinear manifolds. *TIP*, 24(6):1839–1851, 2015.
[51] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, pages 1–14, 2014.
[52] J. Song, Y. Yang, Z. Huang, H. T. Shen, and J. Luo. Effective multiple feature hashing for large-scale near-duplicate video retrieval. *TMM*, 15(8):1997–2008, 2013.
[53] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, 2014.
[54] A. Vedaldi and K. Lenc. Matconvnet: Convolutional neural networks for matlab. In *ACM MM*, pages 689–692, 2015.
[55] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, pages 3424–3431, 2010.
[56] J. Wang, J. Sun, J. Liu, X. Nie, and H. Yan. A visual saliency based video hashing algorithm. In *ICIP*, pages 645–648, 2012.
[57] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.
[58] F. Wu, Z. Yu, Y. Yang, S. Tang, Y. Zhang, and Y. Zhuang. Sparse multi-modal hashing. *TMM*, 16(2):427–439, 2014.
[59] X. Xu, F. Shen, Y. Yang, and H. T. Shen. Discriminant cross-modal hashing. In *ACM ICMR*, pages 305–308, 2016.
[60] Z. Xu, Y. Yang, and A. G. Hauptmann. A discriminative cnn video representation for event detection. In *CVPR*, pages 1798–1807, 2015.
[61] G. Ye, D. Liu, J. Wang, and S.-F. Chang. Large-scale video hashing via structure learning. In *ICCV*, pages 2272–2279, 2013.
[62] H. Ye, Z. Wu, R.-W. Zhao, X. Wang, Y.-G. Jiang, and X. Xue. Evaluating two-stream cnn for video classification. In *ICMR*, pages 435–442, 2015.
[63] L. Yu, Z. Huang, J. Cao, and H. T. Shen. Scalable video event retrieval by visual state binary embedding. *TMM*, 18(8):1590–1603, 2016.
[64] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, pages 4694–4702, 2015.
[65] X. Zeng, W. Ouyang, M. Wang, and X. Wang. Deep learning of scene-specific classifier for pedestrian detection. In *ECCV*, pages 472–487. 2014.
[66] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *TIP*, 24(12):4766–4779, 2015.
[67] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, pages 1556–1564, 2015.
[68] J. Zhou, G. Ding, and Y. Guo. Latent semantic sparse hashing for cross-modal similarity search. In *ICMR*, pages 415–424, 2014.
[69] W. Zhou, M. Yang, H. Li, X. Wang, Y. Lin, and Q. Tian. Towards codebook-free: Scalable cascaded hashing for mobile image search. *TMM*, 16(3):601–611, 2014.
[70] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. In *AAAI*, pages 2415–2421, 2016.