Deep Video Code for Efficient Face Video Retrieval

Shishi Qiao[1,2], Ruiping Wang[1,2,3], Shiguang Shan[1,2,3], Xilin Chen[1,2,3]

[1] Key Laboratory of Intelligent Information Processing of Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing, 100190, China
[2] University of Chinese Academy of Sciences, Beijing, 100049, China
[3] Cooperative Medianet Innovation Center, China
`shishi.qiao@vipl.ict.ac.cn`, {`wangruiping, sgshan, xlchen`}`@ict.ac.cn`

Abstract. In this paper, we address the problem of face video retrieval. Given one face video of a person as query, we search the database and return the most relevant face videos, *i.e.*, ones have same class label with the query. Such problem is of great challenge. For one thing, faces in videos have large intra-class variations. For another, it is a retrieval task which has high request on efficiency of space and time. To handle such challenges, this paper proposes a novel Deep Video Code (*DVC*) method which encodes face videos into compact binary codes. Specifically, we devise a multi-branch CNN architecture that takes face videos as training inputs, models each of them as a unified representation by temporal feature pooling operation, and finally projects the high-dimensional representations into Hamming space to generate a single binary code for each video as output, where distance of dissimilar pairs is larger than that of similar pairs by a margin. To this end, a smooth upper bound on triplet loss function which can avoid bad local optimal solution is elaborately designed to preserve relative similarity among face videos in the output space. Extensive experiments with comparison to the state-of-the-arts verify the effectiveness of our method.

1 Introduction

Face video retrieval in general is to retrieve shots containing a particular person given one video clip of him/her [1]. As depicted in Fig. 1, given one face clip as query, we search the database and return the most relevant face clips according to their distance to the query. It is a promising research area with wide range of applications, such as: 'intelligent fast-forwards' - where the video jumps to the next shot containing the specific actor; retrieval of all the shots containing a particular family member from thousands of short videos; and locating and tracking criminal suspects from masses of surveillance videos [2].

While face video retrieval is in great demand, it is still of great challenge. The video data usually tends to have unconstrained recording environment, which leads to large intra-class variations caused by illumination, pose, expressions, resolution and occlusion, as is shown in Fig. 1. Fortunately, videos provide multiple consecutive faces of one person and each frame forms a part of the person's

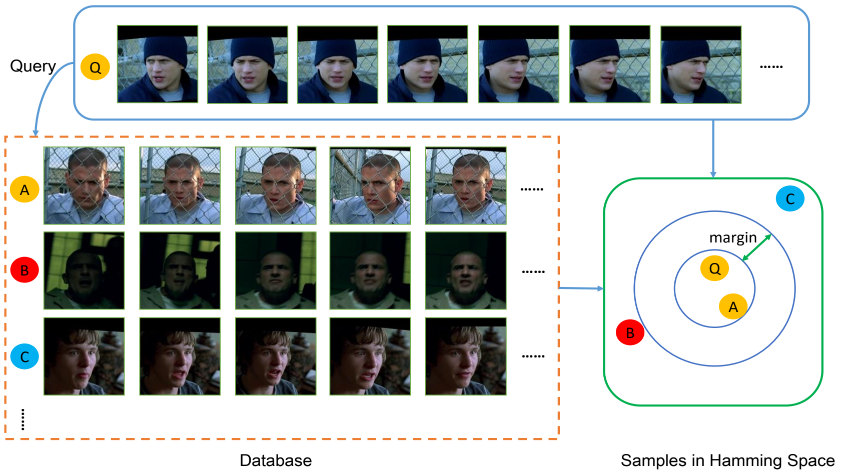


Fig. 1. Illustration of face video retrieval and the motivation of our method. Q, A, B and C denote face clips and the colors around them represent their class labels. Given the query Q, we search the database to find the most relevant face clips according to their distance to Q. It is observed that Q, A, B and C have large intra and inter class variations caused by illumination (B), pose (Q), expressions (C), occlusion (A), etc. We aim to project them into a common Hamming space where distance of dissimilar pairs is larger than that of similar pairs by a margin.

story. One can mine complementary information from each frame to obtain a comprehensive representation for the video. However, modeling face video as a whole is nontrivial. To address this issue, a typical class of video-based face recognition methods [3–10] put the sequential dynamic information of video aside, and simply treat the video as a set of images (*i.e.* frames) and then formulate the problem as image set classification. While encouraging performance has been gained in such works, most of them utilize hand-crafted features which cannot well capture the semantic information and are hard to deal with the challenging intra-class variations. To overcome such limitations, we devise a multi-branch CNN architecture to model face video as a whole by exploiting the convolutional temporal feature pooling scheme which has been proved effective and efficient for video classification task [11]. Another challenge of face video retrieval task is the demand for low memory cost and efficient distance calculation, especially in large scale data scenarios. Obviously, high-dimensional representation of the video is not the best choice. Here we resort to hashing method which is a popular solution for approximate nearest neighbor search. In general, hashing is an approach of transforming the data item to a low-dimensional representation, or equivalently a short code consisting of a sequence of bits [12]. Therefore, we hope the designed CNN architecture can further project the high-dimensional representation into compact binary code, which we name as Deep Video Code (DVC), to satisfy the aforementioned demand with ease in an end-to-end learning manner.

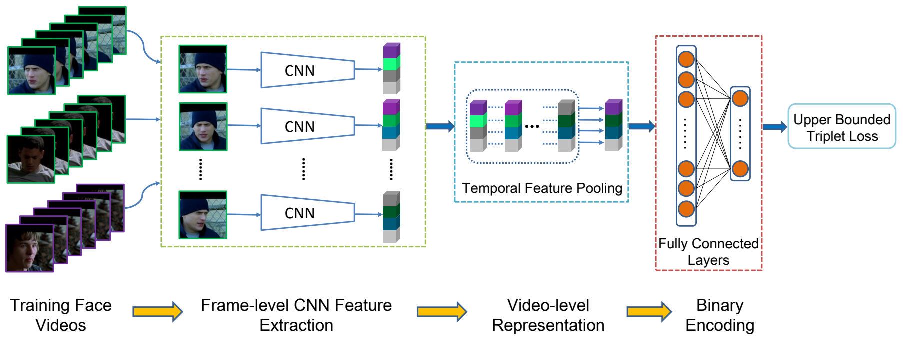


Fig. 2. Framework of the proposed DVC method. We integrate frame-level feature extraction, video-level representation and binary encoding in a unified framework by designing an end-to-end multi-branch CNN architecture. Taking face videos with their class labels as training inputs, DVC first extracts convolutional features for each frame and then utilizes temporal feature pooling on all frames belonging to the same video to produce video-level representation. Finally the fully connected layers project the feature representations of all face videos from the high-dimensional Euclidean space into a much lower-dimensional Hamming space, using the elaborately designed upper bounded triplet loss function.

Fig. 2 shows the framework of our method. The multi-branch network takes face videos instead of face images as training inputs. Then each branch (weights of branches are shared) of the network extracts convolutional features for each frame simultaneously. In order to further obtain a unified representation for each video, we regard the convolution filter as a local concept classifier or detector [13] and pool the classification or detection results of all frames temporally to mine complementary information of frames within one video. With such representation, we further elaborately design a triplet loss function which aims to separate the positive sample pair (a pair of samples coming from the same class) from the negative pair by a distance margin to generate a single binary code for each video as output. To avoid converging to a bad local optimum, the gradients of the loss function should descend stably. To this end, we turn to optimize a smooth upper bound on the loss function inspired by a recent metric learning method [14]. Extensive experiments with comparison to the state-of-the-arts on two challenging TV-Series datasets released in [15] verify the effectiveness of our method on face video retrieval task.

The rest of this paper is organized as follows: We start with related works to our method in Section 2. Section 3 describes DVC in detail. Section 4 evaluates the proposed method with comparison to the state-of-the-arts extensively. Section 5 ends the paper with conclusions.

2 Related Work

As introduced above, our method treats face video as a whole and aims to encode high-dimensional video representations into compact binary codes. Hence, in this section, we give a brief review of related works including face video retrieval, image set and video analysis, and hashing methods.

Face Video Retrieval. More and more works on face video retrieval have been published in recent years [16, 17, 2, 18, 1, 19, 15, 20]. Arandjelović and Zisserman [16, 17] built an end-to-end system to retrieve shots in feature-length films. They proposed a cascade of processing steps to normalize the effects of the changing image environment and use the signature image to represent face shot. However, they did not make full use of information provided by multiple frames. To take advantage of rich information of videos, [2] developed a video shot retrieval system which represents each face video as distributions of histograms and measures them by chi-square distance. These early works utilize high-dimensional features which are not appropriate for efficient retrieval task. [19] is probably the first work which proposed to compress face video into compact binary code by means of learning to hash. [15] further improved the video modeling procedure by representing face video as the set covariance matrix with Fisher Vector as frame feature. [20] proposed a hashing method across Euclidean space and Riemannian manifold to solve the problem of face video retrieval with image query. While certain successes have been achieved in these works, they make little effort to frame-level feature extraction by merely relying on hand-crafted features, which are unfavorable of handling realistic challenging image variations. [21] thus made an early attempt to employ a typical standard deep CNN network to extract frame-level features and hashing codes separately for each single frame. Since their learning totally ignores the correlation information among consecutive frames and the resulting frame-level hashing codes need to be further aggregated to form the final video-level hashing code, the ordinary CNN network framework seems not an optimal solution to efficient video hashing.

Image Set and Video Analysis. Recent years have witnessed an increasing works on video-based face recognition, and among them, a typical class of methods [3–10] simply treat the problem as image set (formed by frames) classification and focus on modeling image set with different representations and measuring their similarities. However, the frame-level features they use lack strong representation power. More recently, deep CNN based methods for video classification [13, 22, 23, 11], event detection [13] and pose estimation [24] have achieved outstanding performance compared with conventional hand-crafted features based methods. [22, 23] proposed to utilize 3D CNN to capture spatial appearance and temporal motion information of video sequence. [13] discovered that simply aggregating the frame-level latent concept descriptors can achieve promising performance. However, the extraction and aggregation of frame-level descriptors are separated which may degrade the performance. To tackle this problem, [11] experimented with different frame-level feature aggregation methods in end-to-end CNNs and found that the performance of simple temporal feature pooling has been quite comparable with other more complex alternatives, verifying the usefulness of joint learning of image representation and feature aggregation.

Hashing Methods. Hashing is widely applied in retrieval area especially for large-scale approximate nearest neighbor (ANN) search problem. Compared with retrieval methods using real-valued features such as [25–27], hashing is more space and time efficient. In early years, studies mainly focus on data-independent hashing methods, such as a family of methods known as Locality Sensitive Hashing (LSH) [28] and random matrix factorization based method [10]. However, LSH methods usually require long codes to achieve satisfactory performance. To overcome this limitation, data-dependent hashing methods attempt to learn similarity-preserving compact binary codes using training data. Such methods can be further divided into unsupervised [29–31] and (semi-)supervised methods [31–44]. Since unsupervised methods cannot take advantage of label information, their performances are usually inferior to supervised methods. In supervised methods, usually an objective function in the form of point-wise [41], pairwise [32–36, 39, 42] or triplet [45, 37, 38, 40, 43, 44] loss is designed. Compared with point-wise and pairwise loss, the objective of triplet loss is to preserve rank order among samples which is very well suited to the preservation of semantic similarity on challenging datasets [45]. In light of the recent progress of deep CNN network in learning robust image representation, there are also growing interests in developing hashing methods using CNN architecture for traditional image retrieval task, such as DLBHC [41] and DNNH [43]. However, as noted in [14], when using the triplet loss function in deep CNN network, it is better to make use of "difficult" triplets and straightforwardly optimizing such triplet loss with mini-batch gradient descent algorithm would probably lead to a bad local optimum [46, 47]. This observation encourages us to elaborately optimize a smooth upper bound on the triplet loss function in our devised multi-branch CNN architecture, which aims to preserve relative similarity among face videos in the output Hamming space and simultaneously avoids bad local optimal solution.

3 Approach

Our goal is to learn compact binary codes for face videos such that: (a) each face video should be treated as a whole, *i.e.*, we should learn a single binary code for each video; (b) the binary codes should be similarity-preserving, *i.e.*, the Hamming distance between similar face videos should be smaller than that between dissimilar face videos by a margin. To fulfill the task, as demonstrated in Fig. 2, our method mainly involves two steps: 1) video modeling, which extracts frame-level features and aggregates them for single video-level representation, and 2) binary encoding, which learns the optimal binary codes for face videos under the designed upper bounded triplet loss function.

3.1 Video Modeling

In this step, what we need is to learn a powerful representation for each face video with the help of CNN. A straightforward method is to train a CNN model first, extract CNN features for each frame with the model, and then aggregate them into a unified representation (e.g. by averaging features of all frames). However, such method separates frame-level feature extraction and video-level representation, which may lead to poor coupling between the two steps and thus poor performance.

Another solution widely used in video modeling with CNNs is 3D Convolution Neural Network (3D CNN). To cope with video data, the 3D CNN takes the motion variation in temporal dimension into consideration. In [22], the 3D convolution is implemented by stacking image frames to construct a cube and then convolving the 3D kernels on the cube. Since 3D CNN connects each feature map to multiple contiguous feature maps (frames for the first convolution layer) in the previous layer, it has the power of making use of more frames' information in videos. However, the goal of 3D CNN is to capture motion information from frame sequences and appearance information from each frame simultaneously, which has high request on the network. Consequently learning of both motion and appearance information is degraded. As far as the face video retrieval task concerns, the temporal motion information among frames contributes little to distinguishing one face video from another for that we care more about the appearance of the faces in video than their heads' motion. Therefore 3D CNN is not the best choice for face video modeling.

For the sake of extracting appearance information and modeling the video as a whole simultaneously, we devise the multi-branch CNN architecture, as illustrated in Fig. 2. Let $F = [f_1, f_2, ..., f_n]$ be a face video with n frames, where f_i denotes the i-th frame. We first propagate each frame through the stacked convolution layers, pooling layers and ReLU [48] non-linear activation layers in one of the multiple branches. By doing this, the CNN features $\mathbf{d}_i \in \mathbb{R}^m$ for each frame are produced. Next, to mine complementary information from these frame-level CNN activations, we adopt the temporal feature pooling methods. Temporal feature pooling has been extensively used for video classification, the resulting vector of which can be used to make video-level predictions [11]. Generally speaking, faces in a video have large variations. Each of them may only carry partial but complementary information. Specifically, the convolution kernel can be regarded as a local concept classifier or detector. While some concepts only exist on one or a few faces in the video, the detectors will only have large responses on some of these faces. By using back-propagation algorithm during training the CNN, gradients coming from the top layers help learn useful local concept detectors, while allowing the network to decide which input frame is relevant to the video representation. In this paper, we test two kinds of temporal feature pooling, *i.e.*, the max-pooling and average pooling.

Until now, we have obtained the m-dimensional vector which serves as the real-valued representation of the face video. In the following we will continue to prorogate the network activations to encode the representation into compact and similarity-preserving binary codes.

3.2 Binary Encoding

We can obtain a representation for each face video using the temporal feature pooling as introduced above. However, such representation suffers from high dimensionality which leads to high space and time complexity when applied to retrieval task straightforwardly. Besides, we still need to devise an objective function to supervise the learning of video representation. To address these problems, we propose a hashing method with triplet loss function. By doing this, the high-dimensional representation is further projected into a much lower-dimensional Hamming space.

To guarantee the similarity-preserving power of learned hashing functions, several kinds of objectives are proposed. Among them, the triplet ranking loss based hash learning methods are very promising because the objective of triplet constraints is to preserve relative rank order among samples, which is agree with the objective of retrieval task. The triplet constraints can be described as the form: "image i is more similar to image j than to image k" [43]. When the dataset is challenging, such constraints are easier to be satisfied than point-wise or pairwise constraints. Apart from that, such form of triplet-based relative similarities are easier to be constructed than others (e.g., for two images with multiple attributes/tags, simply count the number of common attributes/tags as the similarity metric between them). For better understanding of the triplet ranking loss in hash learning, let i, j, k be three samples and i is more similar to j than to k, our goal is to map these three samples into Hamming space where their relative similarity or distance can be well preserved, as illustrated in Fig. 1. Otherwise, punishment should be put on them, defined by:

$$\begin{aligned} J_{i,j,k} = &\max(0, \alpha + D_h(\mathbf{b}_i, \mathbf{b}_j) - D_h(\mathbf{b}_i, \mathbf{b}_k)) \ . \\ &s.t. \ \ \mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k \in \{0,1\}^c \end{aligned} \tag{1}$$

where $D_h(\cdot)$ denotes the Hamming distance between two binary vectors and $\alpha > 0$ is a margin threshold parameter. $\mathbf{b}_i$, $\mathbf{b}_j$ and $\mathbf{b}_k$ are the c-bit binary codes of sample i, j and k, respectively.

Generally we use the gradient descent algorithm with mini-batch to train the CNN with the aforementioned triplet ranking loss. In this case, triplets are constructed at random. Therefore, a substantial part of them contribute little to the convergence of the network during each iteration as they already meet the triplet constraint as described in Eqn.(1) or their loss is quite small. To cope with this problem, our approach tends to make use of "difficult" triplets, *i.e.*, given a pair of similar samples, we actively find the dissimilar neighbor closest to them in current learned Hamming space. Based on this idea, we rewrite Eqn.(1) and give the overall loss function per batch as:

$$\begin{aligned} J = &\frac{1}{2|\widehat{\mathcal{P}}|} \sum_{(i,j)\in\widehat{\mathcal{P}}} \max(0, J_{i,j}) \ , \\ J_{i,j} = &\max\Bigg(\max_{(i,k)\in\widehat{\mathcal{N}}} \{\alpha - D_h(\mathbf{b}_i, \mathbf{b}_k)\}, \\ &\max_{(j,l)\in\widehat{\mathcal{N}}} \{\alpha - D_h(\mathbf{b}_j, \mathbf{b}_l)\} \Bigg) + D_h(\mathbf{b}_i, \mathbf{b}_j) \ . \\ &s.t. \ \ \mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k, \mathbf{b}_l \in \{0,1\}^c \end{aligned} \tag{2}$$

where $\widehat{\mathcal{P}}$ and $\widehat{\mathcal{N}}$ are the set of positive and negative pairs (*i.e.*, similar and dissimilar pairs) in the training mini-batch, respectively. Note that here we allow both sample i and j play the role of anchor point in the triplet structure in order to make full use of samples in the batch.

However, such loss function is non smooth, which causes the network converge unstably and is very likely to get into a bad local optimum. Inspired by [14], we turn to optimize a smooth upper bound on Eqn.(2), defined as:

$$\begin{aligned}
\tilde{J} =& \frac{1}{2|\widehat{\mathcal{P}}|} \sum_{(i,j)\in\widehat{\mathcal{P}}} \max(0, \tilde{J}_{i,j}) \ , \\
\tilde{J}_{i,j} =& \log\Bigg(\sum_{(i,k)\in\widehat{\mathcal{N}}} \exp\{\alpha - D_h(\mathbf{b}_i, \mathbf{b}_k)\} \\
&+ \sum_{(j,l)\in\widehat{\mathcal{N}}} \exp\{\alpha - D_h(\mathbf{b}_j, \mathbf{b}_l)\}\Bigg) + D_h(\mathbf{b}_i, \mathbf{b}_j) \ . \\
&s.t. \ \ \mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k, \mathbf{b}_l \in \{0,1\}^c
\end{aligned} \tag{3}$$

From Eqn.(3), we can see that the triplet loss $\tilde{J}_{i,j}$ takes all dissimilar pairs into consideration which makes the overall loss $\tilde{J}$ for each batch more smooth. At the same time, to make use of "difficult" triplets, the exp operator strengthens the contributions of such "difficult" triplets while weakens other "easy" ones in summation terms.

Unfortunately, it is infeasible to optimize Eqn.(3) directly because the binary constraints require discretizing the real-valued output of the network (e.g. with signum function) and will make it intractable to train the network with back propagation algorithm. For ease of optimization, we adopt the strategy in [43], *i.e.*, replace the Hamming distance $D_h(\cdot)$ with square of Euclidean distance and relax the binary constraints on $\mathbf{b}$ to range constraints. We formulate the relaxed overall loss function as follows:

$$\begin{aligned}
\tilde{J} =& \frac{1}{2|\widehat{\mathcal{P}}|} \sum_{(i,j)\in\widehat{\mathcal{P}}} \max(0, \tilde{J}_{i,j}) \ , \\
\tilde{J}_{i,j} =& \log\Bigg(\sum_{(i,k)\in\widehat{\mathcal{N}}} \exp\{\alpha - D_e^2(\mathbf{b}_i, \mathbf{b}_k)\} \\
&+ \sum_{(j,l)\in\widehat{\mathcal{N}}} \exp\{\alpha - D_e^2(\mathbf{b}_j, \mathbf{b}_l)\}\Bigg) + D_e^2(\mathbf{b}_i, \mathbf{b}_j) \ . \\
&s.t. \ \ \mathbf{b}_i, \mathbf{b}_j, \mathbf{b}_k, \mathbf{b}_l \in (0,1)^c
\end{aligned} \tag{4}$$

where D_e denotes the Euclidean distance and the binary constraints on $\mathbf{b}_i$, $\mathbf{b}_j$, $\mathbf{b}_k$ and $\mathbf{b}_l$ are relaxed to range constraints of 0 to 1.

With Eqn.(4), back-propagation algorithm with mini-batch gradient descent method is applied to train the network. Specifically, we give the gradients of

Eqn.(4) with respect to the relaxed binary vectors as follows:

$$
\begin{aligned}
\frac{\partial \tilde{J}}{\partial D_e^2(\mathbf{b}_i, \mathbf{b}_j)} &= \frac{1}{2|\widehat{\mathcal{P}}|}\mathbb{1}[\tilde{J}_{i,j} > 0] \\
\frac{\partial \tilde{J}}{\partial D_e^2(\mathbf{b}_i, \mathbf{b}_k)} &= \frac{1}{2|\widehat{\mathcal{P}}|}\mathbb{1}[\tilde{J}_{i,j} > 0]\frac{-\exp\{\alpha - D_e^2(\mathbf{b}_i, \mathbf{b}_k)\}}{\exp\{\tilde{J}_{i,j} - D_e^2(\mathbf{b}_i, \mathbf{b}_j)\}} \\
\frac{\partial \tilde{J}}{\partial D_e^2(\mathbf{b}_j, \mathbf{b}_l)} &= \frac{1}{2|\widehat{\mathcal{P}}|}\mathbb{1}[\tilde{J}_{i,j} > 0]\frac{-\exp\{\alpha - D_e^2(\mathbf{b}_j, \mathbf{b}_l)\}}{\exp\{\tilde{J}_{i,j} - D_e^2(\mathbf{b}_i, \mathbf{b}_j)\}} \\
\frac{\partial D_e^2(\mathbf{b}_i, \mathbf{b}_j)}{\partial \mathbf{b}_i} &= 2(\mathbf{b}_i - \mathbf{b}_j), \qquad \frac{\partial D_e^2(\mathbf{b}_i, \mathbf{b}_k)}{\partial \mathbf{b}_i} = 2(\mathbf{b}_i - \mathbf{b}_k) \\
\frac{\partial D_e^2(\mathbf{b}_i, \mathbf{b}_j)}{\partial \mathbf{b}_j} &= 2(\mathbf{b}_j - \mathbf{b}_i), \qquad \frac{\partial D_e^2(\mathbf{b}_j, \mathbf{b}_l)}{\partial \mathbf{b}_j} = 2(\mathbf{b}_j - \mathbf{b}_l) \\
\frac{\partial D_e^2(\mathbf{b}_i, \mathbf{b}_k)}{\partial \mathbf{b}_k} &= 2(\mathbf{b}_k - \mathbf{b}_i), \qquad \frac{\partial D_e^2(\mathbf{b}_j, \mathbf{b}_l)}{\partial \mathbf{b}_l} = 2(\mathbf{b}_l - \mathbf{b}_j)
\end{aligned}
\tag{5}
$$

where, $\mathbb{1}[\cdot]$ is the indicator function which equals 1 if the expression in the bracket is true and 0 otherwise. As is shown in Eqn.(5), the gradients of each iteration contain all negative pairs' information which makes the optimization more stable.

With these computed gradients over mini-batches, the rest of back-propagation can be run in standard manner.

3.3 Implementation details

Network parameters: We implement our DVC method with Caffe platform[1] [49]. Due to memory limitation, we resize each face image to 100×100. In frame-level feature extraction procedure, each branch consists of three convolution-pooling layers. The convolution layers include 32, 32 and 64 5×5 filters with stride 1 respectively, and the size of pooling window is 3×3 with stride 2. Such branch is duplicated multiple times (both the configuration and the weights are shared) and they work side by side. Following the frame-level feature extraction, we implement the temporal feature pooling layer using the Eltwise layer provided by Caffe. Next is two fully connected layers for transforming the high-dimensional representation to low-dimensional real-valued vector. The first fully connected layer contains 500 nodes and the second contains c nodes, where c is the length of the final binary codes. To satisfy the range constraints, we append a sigmoid layer after the last fully connected layer. Moreover, all the convolution layers and the first fully connected layer are equipped with the ReLU activation function.

During training, the weights of each layer were initialized with "Xavier" method[50]. We set batch size to 200, momentum to 0.9 and weight decay to 0.004. We adopt the fixed learning rate policy with 10^{-4} and train the network with $150,000$ iterations. The margin α in Eqn.(4) is empirically set to 1.

[1] The source code of DVC is available at http://vipl.ict.ac.cn/resources/codes.

Training methodology: To speed up the converging of the network, we generate triplet samples online as [14] does, *i.e.*, the input of network is not in triplet form but in point-wise form. Triplets are constructed from each batch at loss layer according to their class labels. By doing this, much more triplets can be utilized during each iteration. As a result, the network will converge faster and the structure of the whole network can be simplified. Therefore the computational resources and storage space can be used more efficiently. However, when distribution of different classes in dataset is uneven, if we draw samples randomly, we may fail to construct a triplet in a mini-batch. Worse still, training in such manner would inevitably cause bad result where classes with large number of samples are trained well and the others may be disappointing. To make sure that face videos of each class distribute uniformly in each batch, we randomly select a few (e.g. 10) class labels first, and then load same number of face videos for each selected class. With such processing, the number of triplets in each batch can be guaranteed and the uneven distribution of different classes as will be introduced in Section 4.1 can be alleviated.

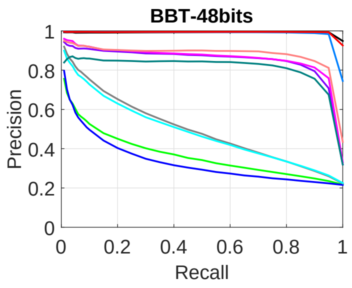
Another trick for training DVC is the use of finetuning. Since the frame-level feature extraction unit of the designed CNN consists of multiple duplicated branches and the convolution operations are time-consuming, training the multi-branch network straightforwardly is not recommended. Hence we turn to first train the network thoroughly with single frame as input (only preserve one branch in frame-level feature extraction unit). Then we expand the single-branch model to the multi-branch model. In this way, we achieve significant speedup compared to training it from scratch. In a similar way, as networks with different code lengths share the same configurations except the last fully connected layer, we train the long code network by finetuning it with short code network. Due to the limited space, evaluation of the effectiveness of finetuning policy on training DVC is introduced in our supplementary materials.

4 Experiments

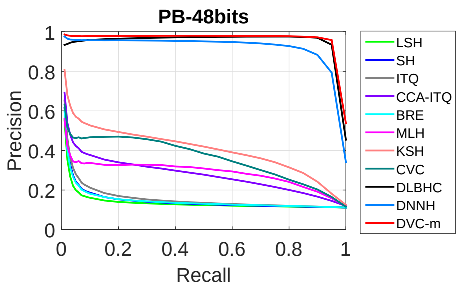
In this section, we first evaluate the power of different CNN architectures discussed in Section 3.1 for video modeling. Then comparison with state-of-the-art hashing methods is conducted to illustrate the effectiveness of the proposed method.

4.1 Datasets and Evaluation Protocols

We carry face video retrieval experiments on the ICT-TV dataset [15]. ICT-TV dataset contains two face video collections clipped from two popular American TV-Series, *i.e.*, the Big Bang Theory (BBT) and Prison Break (PB). These two TV-Series are quite different in their filming style and therefore pose different challenges. BBT is a sitcom and most stories take place indoors. Each episode contains 5$\sim$8 characters. By contrast, many scenes of PB are taken outdoors with a main cast list around 19 characters. Consequently face videos from PB have larger variation of illumination. All the face videos are extracted from the whole



(a) BBT, 48 bits



(b) PB, 48 bits

Fig. 3. Comparison with the state-of-the art hash learning methods with precision recall curves on two databases.

first season of each TV-Series, *i.e.*, 17 episodes of BBT and 22 episodes of PB. Each frame of face video has been cropped to the size of 150×150. The numbers of face videos of these two datasets are 4667 and 9435 respectively. Actually the numbers of face videos of different characters are quite different from each other, which range from 11 to 1528 and 49 to 1965 per character in BBT and PB respectively. To tackle this problem, we adopt the training methodology as introduced in Section 3.3 (due to the limited space, the distribution of face videos per character and validation of the effectiveness of the trick we adopt can be found in the supplementary materials). We use the extracted block discrete cosine transformation features of face images as used in [19] for all traditional methods, *i.e*, methods using hand-crafted features.

We abandon the "Unknown" class in both collections, then randomly select $\frac{2}{3}$ of each character's face videos for training and leave the rest for evaluating the retrieval performance. In this way, we obtain 2971 and 5001 face videos for training, and leave 1487 and 2499 face videos for evaluation on BBT and PB respectively. We will introduce how to use the split for different methods in Section 4.2 and 4.3 in detail.

Following previous works [19, 43, 21], we adopt the mean Average Precision (mAP) and precision recall curves calculated among the whole test set of each dataset for quantitative evaluation.

4.2 Evaluation of Video Modeling

In this part, we validate the effectiveness of the proposed temporal feature pooling. We mainly evaluate different video modeling solutions discussed in Section 3.1, *i.e.*, single-frame model, 3D CNN model, temporal max-pooling model and temporal average pooling model. We denote them as *Single*, *3DCNN*, *T-max* and *T-avg* respectively for convenience. The CNN architectures of the *T-max, T-avg* and *Single* have been discussed in Section 3.3. *3DCNN*'s network configuration is same with *Single* except that the convolution kernels of the first convolution layer are $3 \times 3 \times 3T$ where T is the number of frames of a face video, each frame has 3 channels and totally $3T$ channels in temporal dimension. We can observe that the scale of training set on both dataset is too small to train deep CNNs from scratch. To augment data, we fix the number of frames for each face videos to 10 and segment each face video into multiple sub-videos with such size. Specifically, we slide the segment window 5 frames after each segmentation until the window gets to the last frame. By doing this, we expand the training set from 2971 and 5001 to 25590 and 40941, and the test set from 1487 and 2499 to 12735 and 20357 for BBT and PB respectively. The expanded data is supplied to train *3DCNN*, *T-max* and *T-avg*. For *Single* network, all frames of all face videos in training set are used for training. To represent each face video as a whole, we simply average the representations of the segmented sub-videos or all frames belonging to that face video.

Table 1. Comparison of retrieval mAP of different video modeling methods on BBT and PB with 12-bit binary codes.

	BBT				**PB**			
	3DCNN	*Single*	*T-avg*	*T-max*	*3DCNN*	*Single*	*T-avg*	*T-max*
mAP	0.9759	**0.9853**	0.9791	0.9808	0.8573	0.9547	0.9552	**0.9590**

The retrieval mAP of different models are listed in Table 1. From this table, we can reach three conclusions: (1)*3DCNN* performs worst both on BBT and PB. The reason is that *3DCNN* aims to learn both spatial and temporal information, which has high request on the network. Besides, it mixes appearance information of multiple frames together in the first convolution layer which causes the network fail to capture the appearance information of each frame in the latter layers. Therefore the appearance information of face is not learned well. However, the performance of *Single* method demonstrates the importance of appearance information for face video retrieval. In spite of local motion information, just extracting the appearance information from each single face by *Single* method can achieve promising even best (on BBT) performance. (2) *T-max* and *T-avg* achieve comparable performances with the *Single* method. The reason why *T-max* and *T-avg* perform slight worse than *Single* on BBT is that BBT only has 25590 sub-videos while the number of frames is 136788 for training, and the numbers of network weights in *Single*, *T-avg* and *T-max* are equal. Therefore the sub-videos in training set of BBT for *T-max* and *T-avg* may be not enough. To verify the assumption, we enlarge the training set to 30529 (reduce test set correspondingly) and find that mAP of *T-max* increases to 0.9917 which outperforms *Single* with 0.9892. Besides, the *Single* method needs more computation of projection through fully connected layers for all frames. *T-max* and *T-avg* methods only need to project sub-videos into binary codes, the computation cost of which is much smaller than the former. For PB, sub-videos for training is about 1.6 times as many as that of BBT which is enough to train *T-max* and *T-avg*. Moreover, samples in PB have larger intra and inter class variations than BBT. Hence information carried by different frames is more complementary and the temporal feature pooling operation can mine such information to represent face videos more stably. Therefore, *T-max* and *T-avg* tend to outperform *single* on PB. (3) *T-max* performs better than *T-avg* both on BBT and PB. Proceed from the procedure of convolution operation, each convolutional filter can be regarded as a local concept classifier or detector. When appearance of frames in a video changes fiercely, some local concepts may only exist on one of those faces. Thus, it is better to use the largest activation of convolutional results than the average one.

4.3 Comparison with the State-of-the-art

Comparative methods: We compare our DVC with LSH [28], SH [29], BRE [32], ITQ [31], CCA-ITQ [31], MLH [34], KSH [35], CVC [19], DLBHC [41] and DNNH [43]. Strictly speaking, the ten compared hashing methods except CVC are not specifically designed for face video retrieval task. To conduct face video retrieval experiments on these methods, we trained them by treating a face image as a sample and finally all the frame-level binary representations are fused by hard-voting method as the representation of the face video. For fair comparison, DLBHC and DNNH used the same network structure, as the *Single* network in Section 4.2. Note that in this case, the DNNH actually is the fully connected version described in [43]. For evaluating the loss function, we also implement the *Single* version of DVC, which is named as **DVC-s**. The *T-max* version of DVC is named as **DVC-m**.

Training set: If possible, we would like to use the whole training data to train all methods. However, MLH and KSH cost too much memory. Hence we had to randomly select 5K and 10K frames from all training face videos for MLH and KSH respectively, which costs more than 8GB of memory. Parameters of the compared methods were all set based on the authors' suggestions in their original publications.

Results: Table 2 shows the retrieval performance comparison and Fig. 3 gives the precision recall curves on two datasets with 48-bit binary codes (more results with other code lengths can be found in supplementary materials). In general, supervised hashing methods perform better than unsupervised methods, validating the importance of label information for learning similarity-preserving Hamming space. In addition, those CNN-based methods outperform conventional hashing methods by a large margin, demonstrating the advantage of joint feature learning and binary coding. Apart from that, we also attempted to train some conventional hashing methods with CNN features, their performances improved significantly, but still inferior to our DVC. Detailed results are introduced in supplementary materials.

It is observed that performances of CNN-based methods on BBT are almost the same because face videos in BBT have small variations. However, when it comes to PB which is more challenging than BBT as described in Section 4.1, the performence gap between our method and others especially DNNH becomes larger. We attribute it to the designed loss functions and video representation policies. On one hand, DVC is based on triplet constraints, which aims to optimize the relative rank order among samples. Hence, DVC is very well suited to the preservation of semantic similarity on challenging datasets. Besides, we optimize the smooth upper bound on triplet loss function which biases towards triplets with large loss and simultaneously leads the network to converge stably. However, DLBHC aims to embed label information into binary codes with

Table 2. Comparison of retrieval mAP of our DVC method and the other hashing methods on BBT and PB.

Method	**BBT**				**PB**			
	12-bit	24-bit	36-bit	48-bit	12-bit	24-bit	36-bit	48-bit
LSH [28]	0.2778	0.3062	0.3171	0.3679	0.1259	0.1360	0.1375	0.1412
SH [29]	0.3745	0.3652	0.3473	0.3329	0.1403	0.1496	0.1504	0.1504
ITQ [31]	0.4771	0.4928	0.4924	0.4968	0.1414	0.1525	0.1563	0.1608
CCA-ITQ [31]	0.7159	0.8141	0.8406	0.8547	0.1819	0.2312	0.2595	0.2814
BRE [32]	0.4275	0.4810	0.4869	0.4860	0.1423	0.1468	0.1501	0.1510
MLH [34]	0.7670	0.8058	0.8141	0.8402	0.2294	0.2325	0.2550	0.2783
KSH [35]	0.8819	0.8830	0.8814	0.8856	0.3405	0.3840	0.3993	0.4086
CVC [19]	0.7784	0.8121	0.8158	0.8166	0.2767	0.3314	0.3554	0.3648
DLBHC [41]	0.9870	0.9914	0.9922	0.9922	0.9476	0.9498	0.9521	0.9602
DNNH [43]	**0.9878**	0.9884	**0.9927**	0.9909	0.9262	0.9306	0.9335	0.9262
DVC-s	0.9853	**0.9933**	**0.9927**	**0.9941**	0.9547	0.9663	**0.9785**	**0.9788**
DVC-m	0.9808	0.9926	0.9917	0.9915	**0.9590**	**0.9707**	0.9741	0.9727

point-wise loss which neglects the relative similarity among samples, thus encoding dissimilar images to similar codes would not be punished as long as the classification accuracy is unaffected. Though DNNH is based on triplet constraints, most triplets in a batch contribute little to the convergence of the network for that they already meet the constraints and they overwhelm triplets that violate the constraints. Therefore, our DVC-s outperforms them. On the other hand, DVC-m utilizes the temporal max-pooling network to represent face video which has the ability of making use of more complementary information, while DLBHC, DNNH and DVC-s simply average all frame-level binary codes, the final video code is unreliable when faces have large variations in the video. Moreover, the performance of DVC-m tends to be inferior to DVC-s especially when codes become longer on PB, the reason for it is same with that explained in Section 4.2, *i.e.*, the number of training sub-videos become insufficient when complexity of network becomes high.

5 Conclusion

In this paper, we propose a multi-branch CNN architecture, which takes face videos as inputs and outputs compact binary codes. The learned DVC achieves promising performance compared with state-of-the-art hashing methods on two challenging face video datasets for face video retrieval. We owe it to two aspects: **First**, the integration of frame-level non-linear convolutional feature learning, video-level modeling by temporal feature pooling and hash coding for extracting compact video code. **Second**, the optimization of a smooth upper bound on triplet loss function for hash learning. In the future, we would construct a larger and more challenging face video dataset to train more complicated CNNs.

Acknowledgements. This work is partially supported by 973 Program under contract No. 2015CB351802, Natural Science Foundation of China under contracts Nos. 61390511, 61379083, 61272321, and Youth Innovation Promotion Association CAS No. 2015085.

References

1. Shan, C.: Face recognition and retrieval in video. In: Video Search and Mining. (2010)
2. Sivic, J., Everingham, M., Zisserman, A.: Person spotting: video shot retrieval for face sets. In: Image and Video Retrieval. (2005)
3. Yamaguchi, O., Fukui, K., Maeda, K.i.: Face recognition using temporal image sequence. In: FG. (1998)
4. Cevikalp, H., Triggs, B.: Face recognition based on image sets. In: CVPR. (2010)
5. Hu, Y., Mian, A.S., Owens, R.: Sparse approximated nearest points for image set classification. In: CVPR. (2011)
6. Kim, T.K., Kittler, J., Cipolla, R.: Discriminative learning and recognition of image set classes using canonical correlations. IEEE T PAMI (2007)
7. Wang, R., Chen, X.: Manifold discriminant analysis. In: CVPR. (2009)
8. Wang, R., Shan, S., Chen, X., Gao, W.: Manifold-manifold distance with application to face recognition based on image set. In: CVPR. (2008)
9. Wang, R., Guo, H., Davis, L.S., Dai, Q.: Covariance discriminative learning: A natural and efficient approach to image set classification. In: CVPR. (2012)
10. Parkhi, O., Simonyan, K., Vedaldi, A., Zisserman, A.: A compact and discriminative face track descriptor. In: CVPR. (2014)
11. Ng, J.Y.H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: CVPR. (2015)
12. Wang, J., Shen, H.T., Song, J., Ji, J.: Hashing for similarity search: A survey. arXiv preprint arXiv:1408.2927 (2014)
13. Xu, Z., Yang, Y., Hauptmann, A.G.: A discriminative cnn video representation for event detection. In: CVPR. (2015)
14. Song, H.O., Xiang, Y., Jegelka, S., Savarese, S.: Deep metric learning via lifted structured feature embedding. arXiv preprint arXiv:1511.06452 (2015)
15. Li, Y., Wang, R., Shan, S., Chen, X.: Hierarchical hybrid statistic based video binary code and its application to face retrieval in tv-series. In: FG. (2015)
16. Arandjelović, O., Zisserman, A.: Automatic face recognition for film character retrieval in feature-length films. In: CVPR. (2005)
17. Arandjelović, O., Zisserman, A.: On film character retrieval in feature-length films. In: Interactive Video. (2006)
18. Everingham, M., Sivic, J., Zisserman, A.: Hello! my name is... buffy–automatic naming of characters in tv video. In: BMVC. (2006)
19. Li, Y., Wang, R., Cui, Z., Shan, S., Chen, X.: Compact video code and its application to robust face retrieval in tv-series. In: BMVC. (2014)
20. Li, Y., Wang, R., Huang, Z., Shan, S., Chen, X.: Face video retrieval with image query via hashing across euclidean space and riemannian manifold. In: CVPR. (2015)
21. Dong, Z., Jia, S., Wu, T., Pei, M.: Face video retrieval via deep learning of binary hash representations. In: AAAI. (2016)
22. Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. IEEE T PAMI (2013)
23. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Li, F.F.: Large-scale video classification with convolutional neural networks. In: CVPR. (2014)
24. Pfister, T., Charles, J., Zisserman, A.: Flowing convnets for human pose estimation in videos. In: ICCV. (2015)

25. Chatfield, K., Arandjelović, R., Parkhi, O., Zisserman, A.: On-the-fly learning for visual search of large-scale image and video datasets. International Journal of Multimedia Information Retrieval (2015)
26. Crowley, E.J., Parkhi, O.M., Zisserman, A.: Face painting: querying art with photos. In: BMVC. (2015)
27. Ghaleb, E., Tapaswi, M., Al-Halah, Z., Ekenel, H.K., Stiefelhagen, R.: Accio: A data set for face track retrieval in movies across age. In: ACM on International Conference on Multimedia Retrieval. (2015)
28. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB. (1999)
29. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS. (2009)
30. Liu, W., Wang, J., Kumar, S., Chang, S.F.: Hashing with graphs. In: ICML. (2011)
31. Gong, Y., Lazebnik, S.: Iterative quantization: A procrustean approach to learning binary codes. In: CVPR. (2011)
32. Kulis, B., Darrell, T.: Learning to hash with binary reconstructive embeddings. In: NIPS. (2009)
33. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for scalable image retrieval. In: CVPR. (2010)
34. Norouzi, M., Fleet, D.J.: Minimal loss hashing for compact binary codes. In: ICML. (2011)
35. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: CVPR. (2012)
36. Rastegari, M., Farhadi, A., Forsyth, D.: Attribute discovery via predictable discriminative binary codes. In: ECCV. (2012)
37. Wang, J., Liu, W., Sun, A., Jiang, Y.G.: Learning hash codes with listwise supervision. In: ICCV. (2013)
38. Wang, J., Wang, J., Yu, N., Li, S.: Order preserving hashing for approximate nearest neighbor search. In: ACM International Conference on Multimedia. (2013)
39. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: AAAI. (2014)
40. Zhang, R., Lin, L., Zhang, R., Zuo, W., Zhang, L.: Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. IEEE T IP (2015)
41. Lin, K., Yang, H.F., Hsiao, J.H., Chen, C.S.: Deep learning of binary hash codes for fast image retrieval. In: CVPRW. (2015)
42. Liong, V.E., Lu, J., Wang, G., Moulin, P., Zhou, J.: Deep hashing for compact binary codes learning. In: CVPR. (2015)
43. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: CVPR. (2015)
44. Zhao, F., Huang, Y., Wang, L., Tan, T.: Deep semantic ranking based hashing for multi-label image retrieval. In: CVPR. (2015)
45. Norouzi, M., Fleet, D.J., Salakhutdinov, R.R.: Hamming distance metric learning. In: NIPS. (2012)
46. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: A unified embedding for face recognition and clustering. In: CVPR. (2015)
47. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: BMVC. (2015)
48. Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: ICML. (2010)
49. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: ACM International Conference on Multimedia. (2014)

50. Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: AISTATS. (2010)