Unsupervised Deep Video Hashing via Balanced Code for Large-Scale Video Retrieval

Gengshen Wu, Jungong Han, Yuchen Guo, Li Liu, Guiguang Ding, Member, IEEE, Qiang Ni, Senior Member, IEEE, and Ling Shao, Senior Member, IEEE

***Abstract*—This paper proposes a deep hashing framework, namely Unsupervised Deep Video Hashing (UDVH), for large-scale video similarity search with the aim to learn compact yet effective binary codes. Our UDVH produces the hash codes in a *self-taught* manner by jointly integrating discriminative video representation with optimal code learning, where an efficient alternating approach is adopted to optimize the objective function. The key differences from most existing video hashing methods lie in 1) UDVH is an unsupervised hashing method that generates hash codes by cooperatively utilizing feature clustering and a specifically-designed binarization with the original neighborhood structure preserved in the binary space; 2) a specific rotation is developed and applied onto video features such that the variance of each dimension can be balanced, thus facilitating the subsequent quantization step. Extensive experiments performed on three popular video datasets show that UDVH is overwhelmingly better than the state-of-the-arts in terms of various evaluation metrics, which makes it practical in real-world applications.**

***Index Terms*—Video Hashing, Balanced Rotation, Similarity Retrieval, Feature Representation, Deep Learning**

I. Introduction

With the fast development of Internet and mobile communication technologies, recent decades have witnessed the explosive growth of massive online information. According to the recent statistical reports, for the famous image sharing website, *Flickr*, which contains over 5 billion images, the active users upload over 3,000 images per minute on their personal accounts. While for another video website, *Youtube*, the users post over 100 hours of video clips per minute in the broadcast channels [1]. This deluge of data makes fast similarity search extremely important, however, traditional retrieval methods using exhaustive linear scanning are not feasible in this context due to the critical issues of computational complexity and storage requirement [2]. Consequently, Approximate Nearest Neighbor (ANN) search based on hashing techniques, which represents data by a sequence of binary codes, has attracted substantial attentions attributed to high calculation efficiency with low memory requirement of the XOR operation in the Hamming Space [3], [4], [5].

At the core of hashing-based visual search is how to generate a compound hash function that is able to project high-dimensional floating-point features into the compact binary Hamming space with vital properties from the original data preserved. According to the prior arts, generating binary codes directly from the original feature is usually a NP-hard problem [6], [7], [8], [9]. Consequently, most existing methods adopt a two-stage framework, consisting of projection stage and quantization stage [10]. At the projection stage, certain linear projection functions, which are usually learned from the original data to preserve important properties, such as global Euclidean structure or manifold structure, are built and such functions, in turn, are exploited to project the data from the original feature space to a low-dimensional compact space [11], [12]. At the quantization stage, the real-value feature representation on that space is quantified into binary codes by arbitrary thresholding [3], [13]. With effective projections, such a framework has achieved promising results to some extent. Unfortunately, it has been acknowledged that the performance of those hashing schemes may degrade significantly if the projected dimensions are imbalanced [4], [6], [13], i.e., their variances vary a lot. The major reason is that the equal-length bit allocation adopted in the quantization stage (e.g., 1 bit in most cases, and 2 bits in a Double-bit Quantization [13] ends up with a suboptimal situation that dimensions with lower variances, which contain less information, will have the same influence on Hamming distance computing as high-variance dimensions. The problem mentioned above is illustrated in Fig. 1(a), taking the 2-bit quantization in the two-dimensional feature space as an example. As can be seen, the X axis of the data point obviously contains more information than the Y axis in the original feature space, but both of them are quantized with 1 bit in the Hamming space. This implies that those dimensions containing various amounts of information equally contribute to the calculation of Hamming distance, which is likely to make the quantization intractable.

Recent studies have shown that the spatio-temporal features combining the frame-level spatial information and the temporal information of a video sequence make great contribution in boosting the expressive capability of video representation [14], [15], [16]. Currently, the methods that generate deep video features basically follow two pathways: 1) feeding the sparsely-sampled frame-level image features from video clip into Recurrent Neural Network (RNN) or Long-Short

Manuscript received February 16, 2018; revised July 15, 2018 and October 3, 2018; accepted November 2, 2018. This research was partially supported by the Royal Society Newton Mobility Grant (IE150997), Shenzhen Government (GJHZ20180419190732022), and National Natural Science Foundation of China (Grant No. 61773301, 61571269). (G. Wu and J. Han contributed equally to this work) (Corresponding author: Guiguang Ding).

Gengshen Wu, Jungong Han and Qiang Ni are with the School of Computing and Communication, Lancaster University, Lancaster, LA1 4YW, UK (e-mail: gengshen.wu@lancaster.ac.uk; jungonghan77@gmail.com; q.ni@lancaster.ac.uk).

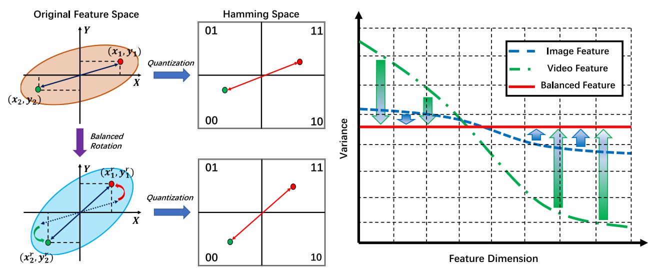
Yuchen Guo and Guiguang Ding are with the School of Software, Tsinghua University, Beijing, 100084, P. R. China (e-mail: yuchen.w.guo@gmail.com; dinggg@tsinghua.edu.cn).

Li Liu and Ling Shao are with Inception Institute of Artificial Intelligence, Abu Dhabi, UAE (e-mail: liuli1213@gmail.com; ling.shao@ieee.org).

Term Memory (LSTM) [17] in order to explore the temporal nature and then aggregating into the global video-level feature [1], [18], [19]; 2) fusing the spatial and temporal features (e.g., optical flow) from the two-stream networks to generate the unified video representation via various pooling schemes [20]. However, we argue that those methods will yield the imbalanced video representation inevitably due to the following reasons. Firstly, the sparse-sampling strategy is usually adopted when tackling video representation learning to reduce the training costs [21]. The contents within those frames are less correlated, especially for the cases of long videos, implying the features in terms of distributions are quite different [22], [23]. Therefore, the imbalanced variance distributions within dimensions can be accumulated when fusing those frame features directly in the video representation learning. There is no evidence that those temporal encoders (e.g., RNN or LSTM) can alleviate such problem. Moreover, the imbalanced situation is even getting worse when making the video-level evaluation with the aggregation of spatial and temporal features in most two-stream based works, which leads to huge variance fluctuations within dimensions. A reasonable explanation is that the temporal network is actually designed to capture different visual aspects of videos compared to the spatial network, which indicates that the feature distributions are more diverse between those fields (e.g., optical flow and RGB) [20], [24], [25]. The arbitrary pooling scheme will aggravate the problem of imbalanced video features. Here, we plot the curves of variances within each dimension of image and video features on ActivityNet [26] in Fig. 1(b) to evidentiate the above discussions, where image features are extracted via a spatial ConvNet [20] and video features are generated via fusing the output vectors of two sub ConvNets in temporal segment networks [20]. As can be observed from Fig. 1(b), the large variance variations do exist within each dimension of video features. However, most existing video hashing methods usually concentrate on selecting the appropriate fusion strategies that can wrap frame-level features up as a single video feature so that the binarization tactics existed in image hashing frameworks can be applied directly [1], [18], [20], [27], regardless of the imbalanced features. This would considerably degrade the quality of video hash codes.

In view of the above analysis, it is clear that such imbalanced problem needs to be addressed carefully in designing video hashing method [28], because applying image feature binarizations to video features directly is suboptimal in producing effective video hash codes. To the best of our knowledge, this is the first attempt to tackle the imbalanced problem when binarizing *video* features. Specifically, we propose a novel hashing framework as shown in Fig. 2, called **U**nsupervised **D**eep **V**ideo **H**ashing (**UDVH**), which distinguishes from the existing methods in three main aspects:

- An unsupervised deep hash framework is proposed to organize the hash code learning in a self-taught manner. Instead of minimizing feature reconstruction distortion [29], our framework minimizes the quantization error of projecting video features to a binary hypercube, thus allowing the feature extraction and hash function learning to engage with each other. Involving the feature clustering in the code learning enables the neighborhood structure to be preserved. To solve the objective function, a novel scheme is proposed, where the rotation matrix, binary code generation and the deep framework parameters are jointly optimized.
- During the code learning, a balanced rotation designed for video features is proposed to identify a proper projection matrix such that the variance of each projected dimension can be balanced (see Fig. 1). By doing so, the information in each dimension of video features can be equalized. This would greatly benefit the quantization step, in which each dimension is allocated with the same number of bits.
- Comprehensive experimental study has been carried out on several publicly available video datasets. The results demonstrate various advantages of UDVH, compared to existing video hashing approaches.



(a) (b)

Fig. 1: (a) Suppose that two data samples (red and green) from a benchmark are projected into a two-dimensional feature space with the coordinates of (x_1, y_1) and (x_2, y_2) and encoded by two bits subsequently, where $|x_1| > |y_1|$, $|x_2| > |y_2|$ and $|x_1 - x_2| > |y_1 - y_2|$. After the proposed balanced rotation, the coordinates of two data points change to (x_1^r, y_1^r) and (x_2^r, y_2^r) accordingly, where $|x_1^r| = |y_1^r|$ and $|x_2^r| = |y_2^r|$. Obviously, compared to the original features, the variances contained in the X and Y axis can be balanced with such rotation strategy applied. That indicates two dimensions of the rotated data points will have the same impact on the calculation of the Hamming distances when encoding them with the fixed number of bits; (b) The variance within each dimension of image and video feature.

This paper is built upon our previous conference paper[1] [19], but it differs in many ways from that work, where the main extensions are summarized in the following list:

- We replace the vanilla stacked LSTM units with Temporal Segment Networks (TSNs) [20] for better feature modeling, which improves the retrieval performance dramatically and validates the powerful scalability of the proposed framework.
- An in-depth analysis of the proposed framework is provided, especially on the comparison between the balanced rotation and other related works.
- We conduct extensive experiments on the proposed framework with TSNs and evaluate the performance un-

[1]We denote the framework in [19] as UDVH-LSTM in this paper.

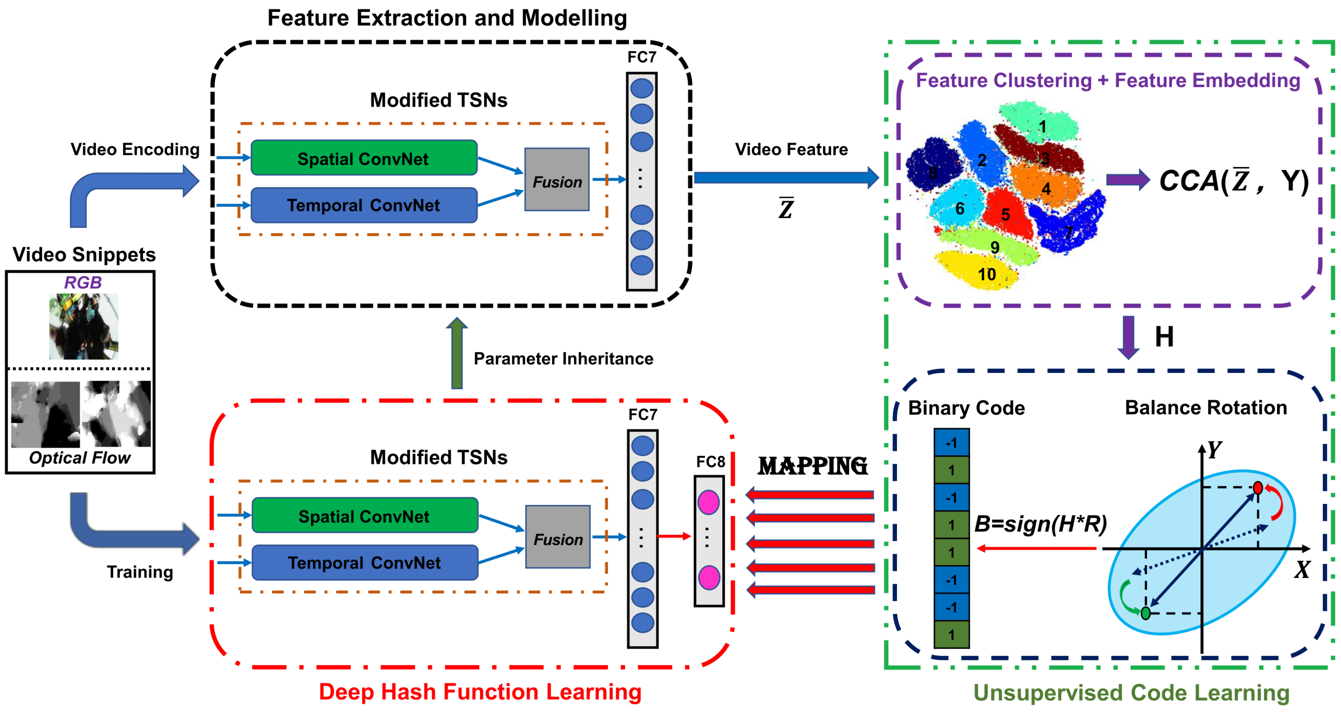


Fig. 2: The basic framework of UDVH-TSN. The whole process consists of three subsections: feature extraction, unsupervised code learning and deep hash function learning, which are performed iteratively to obtain the solution.

TABLE I: The network configurations of SSTH, UDVH-LSTM and UDVH-TSN.

Method	Network Configuration
SSTH [29]	CNN (VGG-19 [30]), encoder RNN including BLSTM and vanilla LSTM layer, decoder RNN including 2-layer stacked vanilla LSTM units
UDVH-LSTM [19]	CNN (VGG-19 [30]), 2-layer stacked vanilla LSTM units, fusion layer, 2 fully-connected layers
UDVH-TSN	Modified TSNs including spatial and temporal ConvNets, fusion layer, 2 fully-connected layers

der various types of loss functions in Section IV. Results on a new dataset, ActivityNet [26], are also provided.

- The experimental results reveal that our new proposed algorithm outperforms the previous work [19], and it is overwhelmingly better than the other state-of-the-art video hashing algorithms.

The rest of this paper is organized as follows: Section II presents the review and discussion of hashing scheme development, including the latest research efforts in video hashing. Section III introduces the proposed UDVH. The experimental results are given and analyzed comprehensively in Section IV. The paper is concluded with detail summary and introduction of future work in Section V.

II. Related Work

In this section, we review and discuss the existing methods closest to our work, including both image and video hashing.

A. Learning-based Hashing

Traditional learning-based hashing methods can be categorized into two independent classes: unsupervised and supervised [31]. For supervised hashing, dedicated label information is utilized for training purpose in the hash function learning. For example, kernel mapping is employed and pair-wised information is utilized by Kernel Supervised Hashing (KSH) [32], where the distances between similar and dissimilar pairs are minimized and maximized, respectively. In [33], LDA hashing is proposed to minimize the intra-class variations and maximize the inter-class variations of binary codes. In unsupervised learning, the hash function that encodes data points into binary codes is built on the training of unlabeled data. Iterative quantization (ITQ) is presented in [3], which minimizes the binarization loss simultaneously by the pre-trained rotation matrix, while an updated version of iterative quantization (ITQ+) is proposed in [11] with both robustness and generalization enhanced by using a l_p-norm distance. Moreover, spectral hashing (SP) is proposed in [7] to obtain binary codes efficiently by solving spectral graph partitioning problem. In [34], K-means hashing (KMH) generates effective hash code via minimizing the Hamming distance between anchors and cells after quantization. The similar idea of using anchors is applied to estimate the similarity between data in Anchor Graph Hashing (AGH) [6]. Recently, breakthrough performance has been achieved by the combination of deep learning techniques and hashing in large-scale image retrieval tasks [35], [36], [37], where deep Convolutional Neural Network (CNN) [38], [39] is widely deployed in those works. In [37], CNNH is proposed to decompose the hash function learning into two stages: hash codes learning and deep network fine-tuning. While DNNH in [36], which can be treated as updated version of CNNH, outperforms CNNH by jointly combining feature learning and hash code learning via deep network. In [35], deep hashing (DH) is developed to learn the hash function with multiple hierarchical non-linear transformations, where independent bits in binary codes with even distribution can be achieved. Subsequently, an unsupervised learning-based deep hashing framework is proposed in [40] that trains the deep neural network by minimizing the compact real-valued codes and the binary codes. They further extend this work to supervised deep hashing and multi-label supervised deep hashing, which aims at generating more discriminative binary codes with aid of supervision information. In [41], a nonlinear discrete hashing method is proposed for scalable image search, where the binary codes are optimized with discrete quantization and the reconstruction errors are minimized between the learnt binary codes and the original data, respectively.

B. Video Hashing

While hashing plays a crucial role in visual search performance improvement, very limited amount of efforts have been invested to video hashing development. Early exploration in the domain often leverages the core ideas of existing image hashing techniques by processing the frame-level features from the video directly. Multiple Feature Hashing (MFH) [42] and Submodular Video Hashing (Submod) [43] are the most representative ones in those algorithms. The former generates binary codes via multiple types of hand-crafted features, while the latter builds the hash function based on the relevant frames selected from videos. However, hand-crafted frame-level features have tremendous limitations in representing the video comprehensively and the results compromise for the suboptimal compatibility with the code learning process [36], [39], [44]. Later on, the temporal nature over successive frames is becoming more attractive in video representation, including motion trajectory [45] and temporal consistency [46]. The experimental results reveal that such dedicated temporal information, rather than those with spatial features adopted, indeed enhances the video representation capability. For example, a low-rank tensor approximation method is introduced in [47] to model the video clips both in 2-D and temporal evolution in the third dimension, thus producing robust video hash codes. Inspired by the recent performance boost of deep learning based image hashing, deep architectures have been adopted in the latest video hashing frameworks to further improve the retrieval effectiveness. One of the most typical examples is a supervised CNN-based hashing framework [18], namely Deep Video Hashing (DVH), which can generate similar binary codes for videos belonging to the same category by exploiting the discriminative temporal nature of video. However, pairwise information is required to compute hash codes in DVH, which might not be easily obtained when dealing with large-scale retrieval tasks. Meanwhile, inspired by the advance of video internal structure in content modelling, Nonlinear Structural Hashing (NSH) [48] is developed to exploit the nonlinear relationship between videos and structural information between frames via subspace clustering. However, temporal information is completely ignored in NSH. The same problem also occurs in [49], where the proposed Stochastic Multiview Hashing (SMVH) converts multiple types of key-frame features into binary codes by exploring the relationship between the original feature and the approximated hash code. [50] substantially upgrades the work mechanism of SMVH via adopting Student t-distribution and deep neural network in the similarity preservation and hash function learning separately.

Alternatively, Zhang et al. [29] propose a deep encoder-decoder framework called Self-Supervised Temporal Hashing (SSTH), where Binary Long Short Term Memory (BLSTM) unit is designed to directly encode the video features into compact binary codes. SSTH tends to minimize the reconstruction distortion between real-valued feature and binary code, where hash function learning and feature extraction are jointly optimized. However, SSTH suffers from serious efficiency issue due to the involvement of de-binarization and de-LSTM. Moreover, minimizing the reconstruction error does not seem to help preserve the neighbourhood structure of the original data, which is crucial for the accurate similarity search. Despite the potential drawbacks analyzed above, SSTH is viewed as a pioneer work in unsupervised deep video hashing with temporal sequence modeling [19] and a strong competitor in the comparisons with the proposed framework. Extension work of SSTH has been released recently in [51], termed SSVH, which incorporates a hierarchical binary auto-encoder and neighborhood structure in the code learning. One of the main drawbacks is that the similarity matrix is only constructed in the initialization stage without considering the temporal information, which limits the improvement of hash code quality. Additionally, the network structures of UDVH-TSN, UDVH-LSTM and SSTH are briefly presented in Table I, where the latter two methods are specifically designed for deep video hashing and inevitably considered to be the major competitors in the following experiments.

III. Unsupervised Deep Video Hashing

Given a benchmark data set that contains N videos, our goal is to exploit the deep hash function $\mathcal{F}(.)$ that encodes those videos into k-bit binary representation as $\mathbf{B} \in \{-1,+1\}^{N\times k}$. Particularly, the deep networks are treated as to-be-learnt binary encoding functions, defined as $\mathcal{F}(\mathbf{Z};\Theta)$, where $\mathbf{Z}$ represents the input that could be in the form of feature matrix or original images. The hash function is parameterized by network parameters Θ including weights and biases. In this paper, instead of stacked LSTM units [19], TSNs [20] are utilized in the feature modeling, which are illustrated in Fig. 2.

Specifically, TSNs evaluate the video-level representation with the two-stream ConvNets networks [20] consisting of spatial and temporal networks. Here, spatial networks collect the spatial information of each image whereas temporal networks model the relationships of successive frames over time with the optical flow fields provided. By fusing these two types of features, TSNs end up exploring both spatial and temporal information to represent videos. To address the over-fitting problem when training deep convolution neural networks on small datasets, several strategies are adopted in training TSNs: (1) Cross modality Pre-training is introduced in the initializations of two ConvNets, where Pre-trained models on ImageNet and RGB frames are utilized as the initial models for spatial and temporal ConvNets, respectively; (2) Compared to the shallow structure in [52], TSNs adopt the Inception with Batch Normalization (BN-Inception) [53] in the network structure, which speeds up the training process significantly by converting the activations of the mean and variance in each batch into the standard Gaussian distribution. In the meanwhile, a good trade-off between algorithm accuracy and efficiency can be achieved by re-evaluating those values in the certain layers with the proposed partial BN [20]; (3) they employ two new methods: corner cropping and scale jittering, in the data augmentation to avoid the severe overfitting problems in traditional two-stream based networks. To make it compatible with the proposed framework, we modify the traditional TSNs presented in [20], where the last *fc-action* layers of the original TSNs are removed and the pooled features from the *global-pool* layers in spatial and temporal ConvNets are fused averagely. Afterwards, such features are fed to two *fc* layers (FC7 and FC8) like UDVH-LSTM [19], thus constructing the architecture of UDVH-TSN.

As discussed in the previous paragraphs, we define the to-be-learnt deep hash function as $\mathcal{F}(\mathbf{Z};\Theta)$, where the deep parameters including weights and biases from the *fc* layers (FC7 and FC8) and modified TSNs for UDVH-TSN are represented by Θ uniformly for the concise description, $\mathbf{Z}$ denotes the input streams. Those network parameters will be updated automatically during each iteration of hash function learning, thus producing the desirable outputs from the last *fc* layer with the dimension of k after the training process. Finally, the binary representation for the input can be obtained by calculating $sign(\mathcal{F}(\mathbf{Z};\Theta))$.

A. Feature Embedding with Pseudo Labels

At the beginning of the proposed unsupervised code learning, we first roughly estimate the feature distribution via exploring the pseudo labels, which aims at improving the effectiveness of the feature embedding afterwards [54]. Particularly, k-means is adopted to categorize the video features $\overline{\mathbf{Z}} \in \mathbb{R}^{N\times 1024}$ from FC7 layer such that the similar videos tend to be classified into the same category. To accelerate the clustering process in the experiment, we randomly sample 10,000 video features from $\mathbf{Z}$ to generate C centroids, where C is the cluster number. Then a 1-of-C vector (C dimensions with one 1 and $C-1$ 0s) can be assigned to each video by measuring the smallest l_2-norm distance among the corresponding feature and those centroids [55]. Such dedicated pseudo labels $\mathbf{Y} \in \{0,1\}^{N\times C}$ are generated for the whole training set during each iteration, thus preserving the original neighborhood structure to the maximum extent in the following process of the dimensionality reduction.

Subsequently, the dimensionality of the video features $\overline{\mathbf{Z}}$ is reduced into the required code length (k) via Canonical Correlation Analysis (CCA) to obtain the projected video feature matrix $\mathbf{H} \in \mathbb{R}^{N\times k}$, where the correlation between video features and the corresponding pseudo labels are maximized and well preserved in the low-dimensional space. In contrast to Principle Component Analysis (PCA), CCA is more effective in extracting discriminative information with the robustness in anti-noise [3], thus obtaining more discriminative video features after the projection. We denote the projection matrix as $\mathbf{P} \in \mathbb{R}^{1024\times k}$ and $\overline{\mathbf{Z}} \in \mathbb{R}^{N\times 1024}$ as the video features from FC7 layer, where $\mathbf{P}$ can be pre-trained with $\overline{\mathbf{Z}}$ and $\mathbf{Y}$ during CCA. According to the above illustrations, the process of the proposed feature embedding can be simplified as:

$$\begin{aligned} &\mathbf{H} = \overline{\mathbf{Z}} \times \mathbf{P} = \overline{\mathbf{Z}} \times CCA(\overline{\mathbf{Z}}, \mathbf{Y}), \\ &\text{s.t. } \overline{\mathbf{Z}} \in \mathbb{R}^{N\times 1024}, \mathbf{Y} \in \{0,1\}^{N\times C}. \end{aligned} \tag{1}$$

However, similar to the conventional methods that reduce the feature dimensionality via projection schemes, CCA will also concentrate most information on a few top eigenvectors, thus unbalancing the projected data and lowering the hash code quality drastically [4]. Consequently, we propose a novel rotation matrix to alleviate this imbalanced issue, which will be elaborated in the next subsection.

B. Balanced Rotation

As shown in Fig. 1(b), the video features are more dispersed in terms of distribution compared to image features, which results in larger variance fluctuations among dimensions. However, in most previous works that allocate the same number of bits to encode such imbalanced features, the dimensionality containing less information (low variance) will make the same contribution on calculating the Hamming distance as those with rich information (high variance), thus significantly reducing the quality of the hash code [4], [6], [13]. To solve the problem described in Fig. 1(a), this paper proposes a novel balanced rotation that balances the information within each dimension of video features before undergoing the quantization. The detailed formula for the proposed rotation is presented as follows. Firstly, the effect of rotation on the variance of data is investigated. Given the optimal projection $\mathbf{P}$, the projected data $\mathbf{H} = \overline{\mathbf{Z}}\mathbf{P} \in \mathbb{R}^{N\times k}$ can be calculated by Eq. (1). Assuming the video features are zero-centralized, i.e., $\sum_{i=1}^{N} \overline{\mathbf{Z}}_i = 0$, the variance of the j-th dimension is $v_j = \frac{1}{N}\sum_{i=1}^{N} \mathbf{H}_{ij}^2$. Given an orthogonal rotation matrix $\mathbf{R}$ and the adjusted data $\mathbf{H}_r = \mathbf{H}\mathbf{R}$, the new variance of each dimension is updated as $v_j{}'$, where $\mathbf{R}\mathbf{R}^T = \mathbf{I}_k$ and $\mathbf{I}_k \in \mathbb{R}^{k\times k}$ is the identity matrix. Then we obtain the following lemma.

Lemma 1: *The sum of variances on all dimensions is invariant after rotation for the centralized data.*

Proof to Lemma 1: Particularly, the sum of variances in all dimensions after rotation can be calculated as below:

$$\begin{aligned} N\sum_{j=1}^{k} v_j{}' &= \sum_{j=1}^{k}\sum_{i=1}^{N} (\mathbf{H}\mathbf{R})_{ij}^2 = tr(\mathbf{H}\mathbf{R}\mathbf{R}^T\mathbf{H}^T) \\ &= tr(\mathbf{H}\mathbf{H}^T) = \|\mathbf{H}\|_F^2 = \sum_{j=1}^{k}\sum_{i=1}^{N} (\mathbf{H})_{ij}^2 = N\sum_{j=1}^{k} v_j, \end{aligned} \tag{2}$$

where $tr(.)$ denotes the trace norm. As observed from the above equation, the total variance remains unchanged with the original properties (e.g., global Euclidean or local manifold structure) preserved after the orthogonal rotation [3], [56]. The purpose of the proposed rotation scheme is to balance the variance of each dimension in the rotated data $\mathbf{H}_r$, where the degree of balance can be measured by the variance of standard deviation (VSD) of each dimension. *Theoretically, smaller VSD implies more balanced data.* If all dimensions have the same data variance or standard deviation, the VSD is 0. Denote the standard deviation (SD) of the j-th dimension as $s_j = \sqrt{v_j}$, the VSD is computed as:

$$\text{VSD} = \frac{1}{k}\sum_{j=1}^{k}(s_j - \overline{s})^2, \tag{3}$$

where $\overline{s}$ is the mean of SD. Hence, the goal of balancing the variances can be achieved by finding a rotation that minimizes the VSD. However, it is not intuitive by solving Eq. (3) directly. This problem can be simplified because of:

Lemma 2: *Minimizing the VSD of the data by a rotation is equivalent to maximizing the sum of standard deviation (SSD).*

Proof to Lemma 2: Based on the Lemma 1, we have $\sum_{j=1}^{k} s_j^2 = \sum_{j=1}^{k} v_j = c$, where c is a constant. Then, the VSD of the data in Eq. (3) can be further expanded as:

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

From Lemma 2, it is clear that minimizing VSD equals maximizing SSD, where the SSD can be further expressed as the matrix form:

$$\text{SSD} = \sum_{j=1}^{k} s_j = \sum_{j=1}^{k}\sqrt{\frac{1}{N}\sum_{i=1}^{N}\mathbf{H}_{ij}^2} = \frac{1}{\sqrt{N}}\|\mathbf{H}^T\|_{2,1}, \tag{5}$$

where $\|.\|_{2,1}$ is the $l_{2,1}$-norm of $\mathbf{H}^T$. Considering Eq. (2), Eq. (4) and Eq. (5) jointly, the rotation matrix which aims at balancing the variance of each dimension can be learned from the elegant optimization formulation:

$$\max_{\mathbf{R}}\|\mathbf{R}^T\mathbf{H}^T\|_{2,1}, \text{ s.t. } \mathbf{R}\mathbf{R}^T = \mathbf{I}_k, \tag{6}$$

where $\mathbf{I}_k \in \mathbb{R}^{k\times k}$ is the identity matrix. Eq. (6) is also the objective function of the proposed balanced rotation and the corresponding optimization process regarding $\mathbf{R}$ will be elaborated in the next subsection.

C. Objective Function and Optimization

In this section, we introduce the objective function of UDVH-TSN[2] and its optimization process in details. The core idea behind such self-taught frameworks is that the binary code $\mathbf{B}$ generated by balanced rotation is utilized iteratively to guide the deep hash function learning, where $\mathbf{B}$ is expected to preserve the local structures and balanced properties from the processes of feature embedding and balanced rotation, respectively [48]. For the purpose of sharing those properties within the hash function construction, the learning objective of hash function learning is defined as minimizing the loss between the output of $\mathcal{F}(\mathbf{Z};\Theta)$ and the learnt binary code $\mathbf{B}$, where l_2-norm distance is used as the measurement. The process can be formulated as following:

$$\mathcal{L}_1 = \min_{\Theta,\mathbf{B}}\|\mathcal{F}(\mathbf{Z};\Theta) - \mathbf{B}\|_F^2, \text{ s.t. } \mathbf{B} \in \{-1,+1\}^{N\times k}. \tag{7}$$

Moreover, the balanced rotation is also considered in this case to address the issue of imbalanced variances during the unsupervised code learning, where its learning objective is formulated as Eq. (6):

$$\mathcal{L}_2 = \max_{\mathbf{R}}\|\mathbf{R}^T\mathbf{H}^T\|_{2,1}, \text{ s.t. } \mathbf{R}\mathbf{R}^T = \mathbf{I}_k. \tag{8}$$

Without loss of generality, we can integrate Eq. (7) and Eq. (8) together to express the overall objective function of UDVH following previous hashing frameworks as below:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_1 + \lambda\mathcal{L}_2 &= \min_{\Theta,\mathbf{B},\mathbf{R}}\|\mathcal{F}(\mathbf{Z};\Theta) - \mathbf{B}\|_F^2 - \lambda\|\mathbf{R}^T\mathbf{H}^T\|_{2,1}, \\ \text{s.t. } \mathbf{B} &\in \{-1,+1\}^{N\times k},\ \mathbf{R}\mathbf{R}^T = \mathbf{I}_k,\ \mathbf{H} \in \mathbb{R}^{N\times k},\end{aligned} \tag{9}$$

where λ is the balance parameter between two terms to provide the general expression of the objective function, i.e., Eq. (9). The value of λ is fixed as 1 during the optimization.

With respect to the optimization, instead of optimizing the objective function in a single step, an alternating approach that differs from previous works is proposed in this paper, where those two sub-objective functions, namely $\mathcal{L}_1$ and $\mathcal{L}_2$, are optimized iteratively and the deep hash functions can be built by updating the parameters, including $\mathbf{R}$, $\mathbf{B}$ and Θ, to facilitate the unique self-taught learning process. This enables the interaction between deep feature learning and hash function learning during the optimization procedure so that the parameters of both deep networks and hash function can be jointly optimized. More importantly, our optimization approach transforms the hash function learning into a regression problem with binary code $\mathbf{B}$ as the regression target, thus avoiding the use of a relaxation strategy to solve the non-convex equation with binary constraints in most published works [3], [8], [18], [35], [48], [57], [58]. Following the above descriptions, our optimization of UDVH can be solved by iteratively optimizing $\mathcal{L}_1$ and $\mathcal{L}_2$. The optimization goal of $\mathcal{L}_2$ can be achieved in the following alternating steps.

1) **Update R**. With given video feature $\mathbf{H} \in \mathbb{R}^{N\times k}$, Eq. (8) can be viewed as the orthogonality constrained $l_{2,1}$-norm maximization problem, which can be efficiently solved by the gradient flow method in [59]. Following [59], a feasible set for $\mathbf{R}$ is defined as $\mathcal{M}_k = \{\mathbf{R} \in \mathbb{R}^{k\times k} : \mathbf{R}\mathbf{R}^T = \mathbf{I}_k\}$, which is also called the *Stiefel* manifold. Then the tangent space for $\mathcal{M}_k$ can be formulated as $\mathcal{E}_R = \{\mathbf{E} \in \mathbb{R}^{k\times k} : \mathbf{E}^T\mathbf{R} + \mathbf{R}^T\mathbf{E} = 0\}$. Here, the basic idea is to find an optimal direction in the tangent space of the current point $\mathbf{R}$, then project that direction to the feasible manifold, and replace the current point with the projected one. Finally, a stationary point can be achieved by repeating the above steps iteratively. To optimize the problem

[2]The objective function is exactly the same for both UDVH-LSTM [19] and UDVH-TSN but with slight differences in defining the network parameters $\mathbf{\Theta}$.

Algorithm 1: Unsupervised Deep Video Hashing

```
Input: The input matrix $\mathbf{Z}$; Randomly initialize deep parameters $\Theta$; Initialize $\mathbf{R}_0 = \mathbf{I}_k$ and $r = 0$; Code length $k$;
Output: $\mathcal{F}(\mathbf{Z}; \Theta)$: deep hash function;
 1: for $t = 1$ to $T$ do
 2:   Compute feature matrix $\mathbf{H}_t$ according to Eq. (1);
 3:   repeat
 4:     Compute $\mathbf{D}_r$ by Eq. (14);
 5:     Compute $\mathbf{G}_r$ by Eq. (13);
 6:     Compute $\mathbf{W}_r$ by Eq. (12);
 7:     Compute $\mathbf{Q}_r$ by Eq. (11);
 8:     Compute $\mathbf{R}_{r+1}$ by Eq. (10);
 9:     $r = r + 1$;
10:   until Convergence;
11:   Update rotation matrix $\mathbf{R}_t = \mathbf{R}_{r+1}$;
12:   Update binary code $\mathbf{B}_t$ according to Eq. (15);
13:   Update deep parameters $\Theta_t$ according to Eq. (16);
14:   Until Convergence;
15:   $t = t + 1$;
16: end for
17: return $\mathcal{F}(\mathbf{Z}; \Theta)$;
```

(10), the convergence lemma is illustrated as: *given a point $\mathbf{R}_r$ in the feasible set, the below updating rule will lead to larger value unless it has arrived at a stationary point, namely:*

$$\mathbf{R}_{r+1} = \mathbf{Q}_r \mathbf{R}_r, \tag{10}$$

where $\mathbf{Q}_r$ is the *Cayley* transformation matrix defined as:

$$\mathbf{Q}_r = (\mathbf{I}_k + \frac{\tau}{2}\mathbf{W}_r)^{-1}(\mathbf{I}_k - \frac{\tau}{2}\mathbf{W}_r). \tag{11}$$

In the above equation, $\mathbf{W}$ is the skew-symmetric matrix, which can be calculated with the upgradient $\mathbf{G}$. They are formulated as follows:

$$\mathbf{W}_r = \mathbf{G}_r \mathbf{R}_r^T - \mathbf{R}_r \mathbf{G}_r^T, \tag{12}$$

$$\mathbf{G}_r = -\mathbf{H}_r^T \mathbf{H}_r \mathbf{R}_r \mathbf{D}_r, \tag{13}$$

$$\mathbf{D}_r = diag(\frac{1}{N^{0.5} s_{r1}}, ..., \frac{1}{N^{0.5} s_{ri}}, ..., \frac{1}{N^{0.5} s_{rk}}). \tag{14}$$

Here, the subscript r denotes the r-th iteration. τ is a step size satisfying *Armijo-Wolfe* conditions [60], which only controls the convergence rate in updating $\mathbf{R}$. Usually bigger τ leads to faster convergence. s_{ti} represents the standard deviation of $(\mathbf{HR}_r)_{*i}$ and $\mathbf{D}$ is the diagonal matrix. The above steps are iteratively executed until convergence and the obtained $\mathbf{R}$ is the rotation matrix.

2) **Update** $\mathbf{B}$. After solving $\mathbf{R}$, the variance of each dimension in the projected video feature $\mathbf{H}$ is balanced and consequently the binary code $\mathbf{B}$ can be calculated via thresholding:

$$\mathbf{B} = sign(\mathbf{H} \times \mathbf{R}), \text{s.t. } \mathbf{H} \in \mathbb{R}^{N \times k},\ \mathbf{R} \in \mathbb{R}^{k \times k}. \tag{15}$$

The obtained balanced codes are prepared to be utilized as the learning objective of network training afterwards.

3) **Update** Θ. With fixed $\mathbf{Z}$, $\mathbf{R}$ and $\mathbf{B}$, the optimization of $\mathcal{L}_1$ in Eq. (7) is further derived as below with the only argument Θ:

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

This minimization problem can be solved by fine-tuning the deep network with Stochastic Gradient Descent (SGD) [61] until it gets converged, where the Euclidean loss is minimized via mini-batch back-propagation and the low bound can be found (See IV-C). The network parameter Θ is updated simultaneously with the balanced properties from $\mathbf{B}$ preserved after convergence, such boosting the quality of hash code in the query process. By repeating above steps iteratively, the deep hash function can be built finally after several iterations, where $t = 3 \sim 5$ in this paper. Given a query video $\mathbf{Z}_q$, the hash code can be obtained via compute $sign(\mathcal{F}(\mathbf{Z}_q; \Theta))$. The step by step description of the proposed optimization scheme is summarized in Algorithm 1.

D. Complexity Analysis

The complexity cost of Algorithm 1 basically consists of three parts as discussed in Section III: deep network training, feature embedding and balanced rotation. However, calculating the complexity for the first two terms is not straightforward because the time costs in network training and feature clustering are affected dramatically by the hardware conditions, which will be discussed later in the Section IV. Here, we focus on the optimization complexity in solving the learning objective of balanced rotation. In particular, the optimization starts by solving $\mathbf{R}$ according to Eq. (10)$\sim$(14), in which the time complexity for (14) is $\mathcal{O}(Nk)$ and (13) is $\mathcal{O}(k^3)$, while computing $\mathbf{H}^T\mathbf{H}$ requires $\mathcal{O}(Nk^2)$, which can be pre-computed and remains unchanged with iterations. For Eq. (10)$\sim$(12), the time complexity is the same as $\mathcal{O}(k^3)$. Thus, the overall complexity in updating $\mathbf{R}$ is $\mathcal{O}(Nk^2 + t(Nk + k^3))$ after t-th iteration which is linear to the size of training data. Actually, considering that k and t are usually quite small in the algorithm, optimizing (9) can be highly efficient. In addition, the time complexity is $\mathcal{O}(k^3)$ in (15) in solving $\mathbf{B}$, which indicates the fast speed of the code learning.

E. Discussion

Now, we discuss the connection between BR and some previous works. The first one is Iterative Quantization (ITQ) [3], which finds a rotation to minimize the quantization error. In fact, ITQ shares a very similar formulation to BR. If closely looking at the objective function of ITQ, we have:

$$\begin{aligned} \|\mathbf{B} - \mathbf{HR}\|_F^2 &= \|sgn(\mathbf{HR}) - \mathbf{HR}\|_F^2 \\ &= const - 2\|\mathbf{R}^T\mathbf{H}^T\|_{1,1}. \end{aligned} \tag{17}$$

Hence, the optimization problem of ITQ can be rewritten into the following formulation,

$$\max_{\mathbf{R}} \|\mathbf{R}^T\mathbf{H}^T\|_{1,1}, \text{s.t. } \mathbf{RR}^T = \mathbf{I}_k. \tag{18}$$

Despite the similar formulations, there are several important differences between ITQ and BR. 1) BR focuses on balancing the variance which can explicitly address the imbalance between dimensions. Although ITQ maximizes the total variance of the data in an indirect way via minimizing the quantization errors, the bits are quite imbalanced; 2) ITQ adopts an iterative strategy with two matrix variables for optimization which can only achieve a local optimal while BR has one matrix variable and a more effective optimization algorithm so that it can achieve better solution. The other two related works are Isotropic Hashing (IsoH) [62] and Harmonious Hashing (HamH) [4]. IsoH aims to find a rotation to balance the variance by minimizing the reconstruction error of the co-variance matrix and a diagonal matrix. But the reconstruction on a small covariance matrix seems restricted, so it may be unstable in large-scale and high-dimensional experiments [4] due to the overfitting problem. HamH seeks for a rotation to minimize the distance between the rotated data and a perfectly balanced matrix. However, such strict requirement and its non-iterative optimization algorithm may fail to find a good enough solution as BR which limits its performance, though it may balance the data to some extent. In addition, IsoH is based on PCA projection and is derived from spectral hashing [7]. There is no clue that they have stable results based on other projection learning methods. Recently, bit-independent constraint has been incorporated in the nonlinear discrete optimization, where the l_2-norm distance is minimized between the hash code and the real-valued matrix set [48]. Although the information content is increased inevitably in this case, the quantization errors are accumulated during the iterative optimization and the aforementioned imbalanced variance within each dimension is still an open issue that affects the hash code quality critically.

IV. Experiments

In this section, three large-scale video datasets are adopted in the experiments, while the corresponding parameters settings are introduced carefully. Then systematic evaluation and analysis of the proposed frameworks are illustrated compared with some state-of-the-art video hashing methods, including some supplementary results for UDVH-LSTM [19].

A. Datasets and Experimental Setting

We validate the proposed deep hashing learning framework on several large-scale public datasets for video retrieval, including **FCVID**[3], **YFCC**[4] and **ActivityNet**[5]. The basic information about those datasets are introduced as below.

Fudan-Columbia Video Dataset (**FCVID**) [63] collects 91,223 videos from *Youtube*, which are classified into 239 categories with accurate manual labels. A wide range of generic topics are included in this datasets, such as sports, events and various scenes with the average length of 167 seconds. We randomly select 45,611 videos for the train split in the unsupervised learning process and the rest is used as the test split in the retrieval.

Yahoo Flickr Creative Common (**YFCC**) [64] is one of the largest public video dataset available in real-world, which contains over 0.8M video clips with the average length of 37 seconds as claimed. However, 0.7M videos are downloaded and processed in this experiment except for some corrupted videos and invalid download links. Particularly, we split the dataset into two parts: 0.6M unlabeled videos as the train split in the unsupervised learning and 0.1M labeled videos provided by [29] as the test split in the retrieval, where there are 80 popular scenes collected from the third level of MIT SUN scene hierarchy [65] in the second part.

ActivityNet [26] covers a wide range of complex human activities in the daily living, which contains around 20,000 video clips that classified into 200 categories. Particularly, those videos are split into training, validation and testing sets with 10,024, 4,926 and 5,044 videos individually. Generally we follow the settings of above datasets, where the whole ActivityNet dataset is split into train and test sets averagely, about 10,000 videos for each set. However, for the purpose of making the best of dataset, we use the original testing videos in the unsupervised training for the labels of those are not released and take some instances from training videos to construct the new test set. For all datasets, 1,000 videos randomly picked up from the test split are used as the query instances and others form the gallery in the online retrieval.

B. Baselines

Several existing hashing methods are adopted as baselines in the experiment, which are Anchor Graph Hashing (AGH) [6], Submodular video hashing (SubMod) [43], Iterative Quantization (ITQ) [3] Spectral Hashing (SP) [7], Multiple Feature Hashing (MFH) [42], Deep Hashing (DH) [35], Self-Supervised Temporal Hashing (SSTH) [29] and UDVH-LSTM[6] [19]. Deep Video Hashing (DVH) [18] and Nonlinear Structural Hashing (NSH) [48] are not considered in this case because both of them are supervised methods. For the consistency of comparison, the experiments are carried out with the identical data sets and the best performance is tuned according to parameter settings in their papers.

C. Implementation Details

Without loss of generality, 20 frames are uniformly selected as the representation of video clips for each instance in above datasets. In the feature embedding, 10,000 video features from $\mathbf{Z}$ are randomly picked up and utilized on all datasets. During the code optimization, r and τ are set to 100 and 0.0005. The experiments are conducted using Matlab 2014a on Ubuntu server configured with Intel Core i7-5960X CPU, 64 GB of RAM and TITAN 1080i GPUs. However, there are some minor differences in the implementation processes for network training for UDVH-TSN when compared to UDVH-LSTM [19], which are detailed as follows.

Generally, we follow the parameter settings in the release codes[7] and its paper of temporal segment network [20] with minor adjustments in the network model, which is illustrated in Sec III. Particularly in this case, we utilize the original RGB frames and extract their optical flow[8] [66] images from videos in above benchmark datasets by OpenCV as the input streams to the spatial and temporal networks, where those network

[3]https://http://bigvid.fudan.edu.cn/FCVID/

[4]https://webscope.sandbox.yahoo.com/

[5]http://activity-net.org/

[6]CNN features in UDVH-LSTM and SSTH are replaced with the frame-level features from the pre-trained TSNs in the following experiments.

[7]https://github.com/yjxiong/temporal-segment-networks

[8]We adopt TV-L^1 optical flow algorithm as [20].

TABLE II: mAP@20 of different cluster numbers at 128 bits on three datasets under UDVH-TSN.

Datset	**UDVH-TSN**				
	C=100	C=200	C=400	C=800	C=1600
FCVID [63]	0.362	0.482	**0.504**	**0.504**	0.502
YFCC [64]	0.201	0.274	**0.293**	0.289	0.291
ActivityNet [26]	0.145	0.198	**0.262**	0.257	0.261

models are pre-trained on UCF101 [67] dataset according to [20]. The modified version of Caffe [20] is also kindly provided by the authors to accelerate the training process. It is worth mentioning that we utilize the original RGB and optical flow frames instead of using extracted CNN feature vectors previously in UDVH-LSTM [19] and evaluate the video accordingly by the uniformly-sampled 20 frames following [20] for the purpose of fair comparison in the experiments. In the network training, however, the mini-batch back-propagation is only performed in the FC7 and FC8 layers, where the output numbers are set to 1024 and k. Gaussian distribution and constant are applied for the initialization of their weights and bias, respectively. The base learning rate is set to 10^{-3} with momentum and weight decay tuned as 0.9 and 0.0005. The batch size is fixed to 256. The maximal training iteration is set to $15,000$. Usually, the proposed networks converge within 10 epochs during each loop (t) of the hash function learning.

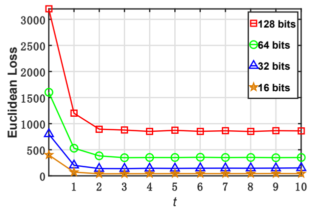
The pooled video features of 1024-d from *global-pool* layers of the pre-trained TSNs models (including spatial and temporal ConvNets) are averagely fused and utilized in evaluating the performance of some baselines, such as AGH, ITQ, SP and DH. While for SSTH and UDVH-LSTM, we extract the frame-level features (i.e., 20 frames uniformly sampled from video clips) from the pre-trained TSNs models and then fuse them frame-by-frame accordingly to obtain the pooled 20 frame features to facilitate their training processes.

D. Evaluation Metrics

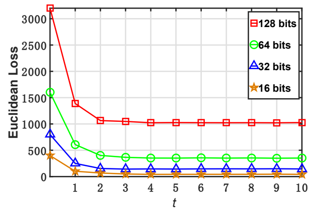
To evaluate the effectiveness of hashing methods, mean Average Precision at top K (mAP@K) retrieved videos is adopted as the main performance metric following [29], which is defined as the mean of average precision of returned relevant videos number in the top K results [63]. Moreover, Precision-Recall (PR) curve and Precision at top 100 (Precison@100) returned samples are adopted as assisted measurements for systematical evaluation [68]. All of those hashing methods are estimated based on four different bit sizes, i.e, 16, 32, 64, 128.

E. Results and Analysis

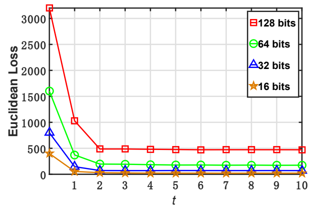
1) Parameter Analysis: We first investigate the influence on retrieval performance when selecting different cluster numbers (C=100, 200, 400, 800, 1600) on three datasets under the frameworks of UDVH-TSN. As shown in Table II, mAP@20 increases dramatically with the bigger cluster number picked up at the beginning and then achieves the best performance when we have about 400 categories. This indicates the cluster number indeed affects the system performance, where the similar videos will be classified into the same category by the sophisticated clustering strategy, thus producing similar hash codes for those videos. It is a remarkable fact that the performance gets saturated when the cluster number reaches 400 for all datasets under both UDVH-LSTM [19] and UDVH-TSN. The reason might be that those datasets contain less categories than 400, which leads to a quick saturation and makes the parameter easy to be configured as well.



(a) FCVID



(b) YFCC



(c) ActivityNet

Fig. 3: The convergence curves on three datasets at various bit sizes under UDVH-TSN, where the final values of Euclidean loss remain stable after $t = 3 \sim 5$ loops.

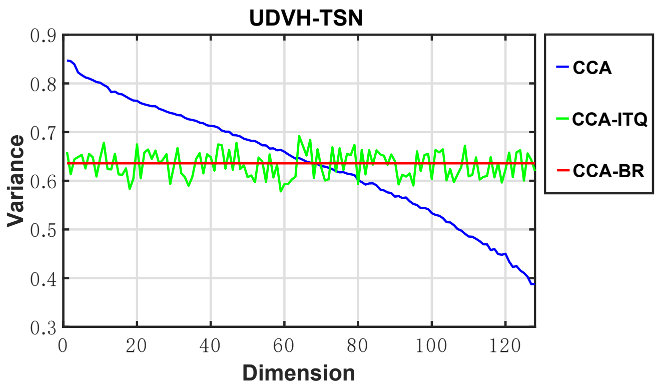


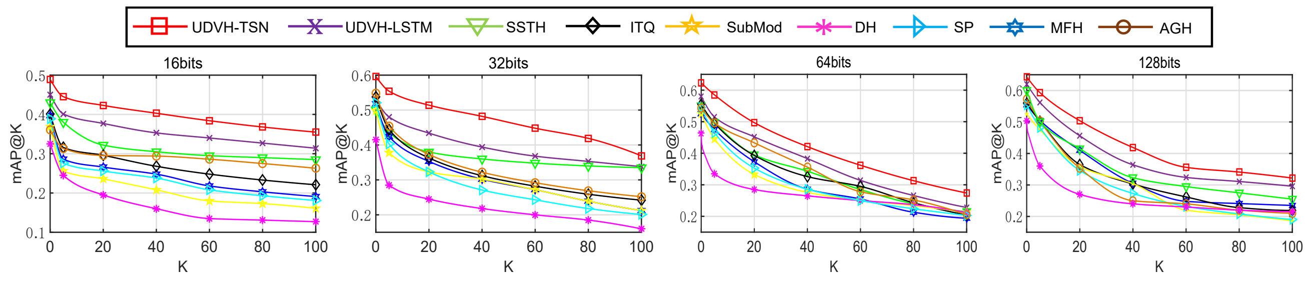
Fig. 4: The variance distribution from FCVID at 128 bits under UDVH-TSN.

2) Convergence Study: Next, we plot the curves of Euclidean loss values after 10 iterations (t) of the hash function learning on three datasets under UDVH-TSN in Fig. 3. Obviously, fast convergence can be easily achieved within about $t = 3 \sim 5$ loops on all datasets, which implies the extremely high efficiency and feasibility of the proposed framework. Combined with the training costs from the following efficiency analysis, it proves that the proposed framework is suitable to be deployed in the large-scale retrieval tasks.

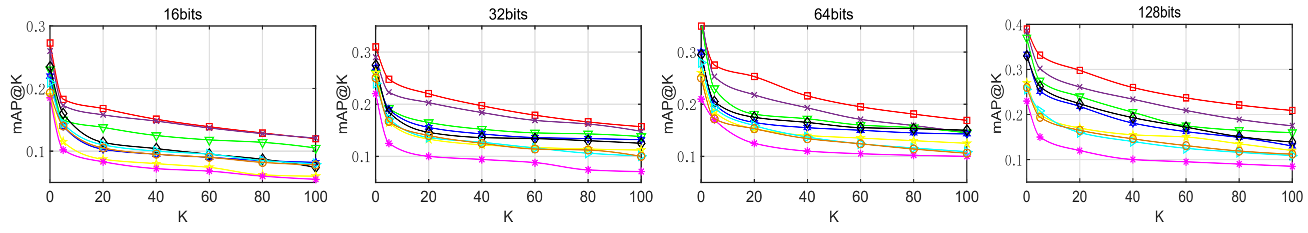
3) Binarization Investigation: In this section, the retrieval performance when using balanced rotation in the binarization is presented carefully, which validates its positive impacts and the claimed contributions. In this experiment, three competitive combinations are adopted: CCA-ITQ, PCA-BR and CCA-BR,

TABLE III: mAP@K of 128 bits when using various banarization schemes separately in the code learning of UDVH-TSN.

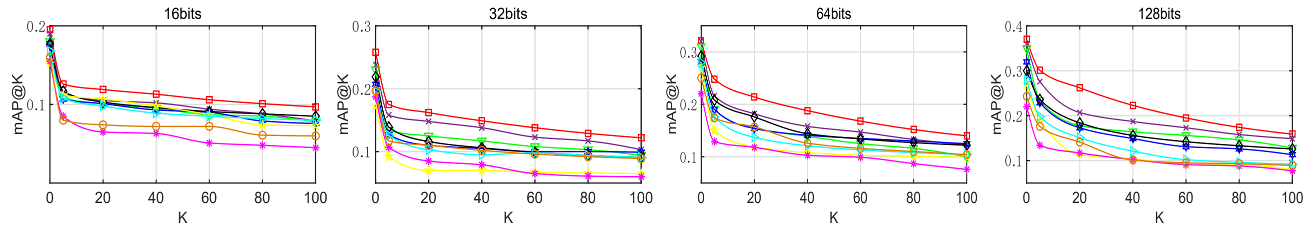
Method	FCVID						YFCC						ActivityNet					
	K=5	K=20	K=40	K=60	K=80	K=100	K=5	K=20	K=40	K=60	K=80	K=100	K=5	K=20	K=40	K=60	K=80	K=100
PCA-BR	0.463	0.366	0.291	0.241	0.205	0.179	0.298	0.258	0.235	0.219	0.198	0.183	0.221	0.192	0.176	0.158	0.148	0.139
CCA-ITQ	0.555	0.487	0.412	0.356	0.303	0.297	0.304	0.276	0.24	0.222	0.207	0.196	0.268	0.231	0.197	0.175	0.168	0.152
CCA-BR	**0.593**	**0.504**	**0.42**	**0.362**	**0.347**	**0.328**	**0.326**	**0.293**	**0.246**	**0.228**	**0.215**	**0.202**	**0.301**	**0.262**	**0.223**	**0.195**	**0.174**	**0.159**



(a) FCVID@16 bits (b) FCVID@32 bits (c) FCVID@64 bits (d) FCVID@128 bits

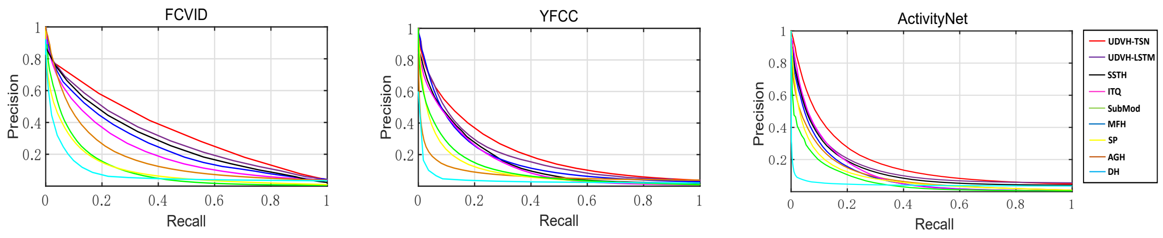


(e) YFCC@16 bits (f) YFCC@32 bits (g) YFCC@64 bits (h) YFCC@128 bits



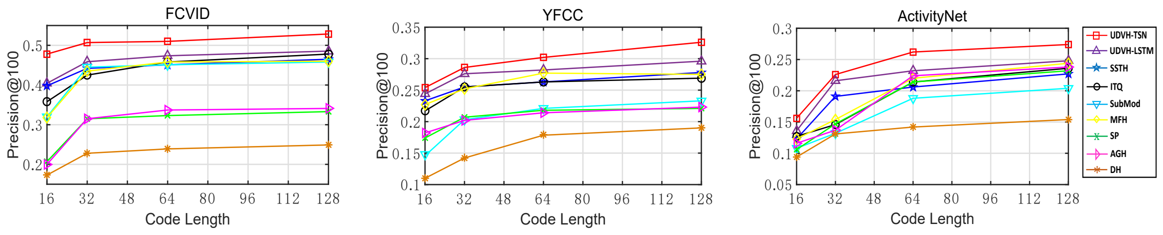
(i) ActivityNet@16 bits (j) ActivityNet@32 bits (k) ActivityNet@64 bits (l) ActivityNet@128 bits

Fig. 5: The mAP@K curves at different bit sizes under UDVH-TSN. Top: FCVID; Middle: YFCC; Bottom: ActivityNet.



(a) FCVID@128 bits (b) YFCC@128 bits (c) ActivityNet@128 bits

Fig. 6: The Precision-Recall curves at 128 bits under UDVH-TSN.



(a) FCVID (b) YFCC (c) ActivityNet

Fig. 7: The Precision@100 curves at various bit sizes under UDVH-TSN.

TABLE IV: mAP@20 at 128 bits when applying various loss functions accordingly during the hash function learning of UDVH-LSTM and UDVH-TSN.

Loss Function	UDVH-LSTM [19]			UDVH-TSN		
	FCVID	YFCC	ActivityNet	FCVID	YFCC	ActivityNet
$l_2 - norm$	**0.456**	**0.261**	**0.207**	**0.504**	**0.293**	**0.262**
$l_1 - norm$	0.442	0.256	0.186	0.483	0.279	0.237
$cross - entropy$	0.447	0.248	0.192	0.501	0.291	0.252

where the mAP@K results are illustrated in Table III when using those binarization strategies on three video datasets separately at the code length of 128.

The results show that the combination of CCA and BR outperforms the other methods on the datasets substantially. Two conclusions can be drawn based on this phenomenon: 1) Compared with the PCA methods, better performance has been achieved by applying CCA in the dimensionality reduction since more valuable information is concentrated and preserved in the top eigenvectors when utilizing the dedicated supervisory information (pseudo labels) from the clustering. It turns out that the choice of CCA is more sensible and effective than PCA; 2) The difference between the results of CCA-ITQ and CCA-BR clearly demonstrates that the retrieval performance can be improved by balancing the variances of video features via the proposed balanced rotation, which indicates the superiority of such framework. In Fig. 4, the variance of each dimension on FCVID at 128 bits by using CCA, CCA-ITQ and CCA-BR is plotted to validate the above conclusions. Again, it is pretty clear that our BR scheme can always guarantee the flat variance of each dimension, regardless of which video features are used.

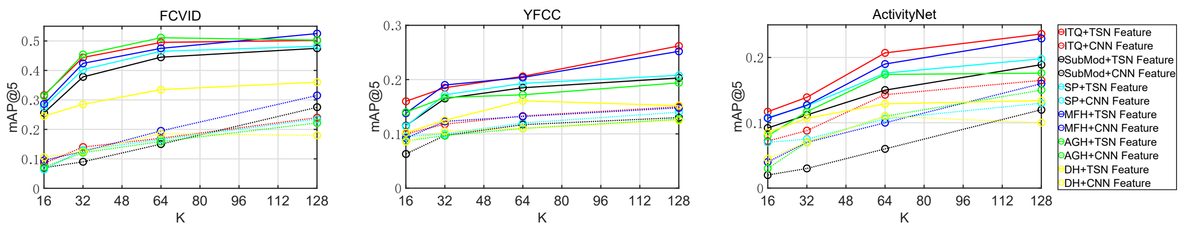
4) Loss Function: As shown in Table IV, we evaluate the system performance under various loss functions: l_2-norm. l_1-norm and cross-entropy, during the network training of UDVH-LSTM and UDVH-TSN. Compared to using l_2-norm in the objective functions, l_1-norm is supposed to be more robust in anti-noising and cross-entropy converts the hash function learning into classification problems. According to the results, however, the effects of using various loss functions are not obvious. Generally, l_2-norm achieves the best performance with limited improvement (less than 3%) on the same dataset.

5) Comparison with State-of-The-Arts: To demonstrate the superiority of our method, we further compare it with some competitive baselines on three large-scale video datasets.

Experiment I on FCVID: We first report the comparison results on FCVID dataset, where Fig. 5(a)$\sim$(d) show the mAP@K curves at various code lengths on the datasets when using different methods separately. As observed from those figures, the proposed method outperforms the state-of-the-art hashing approaches in all cases substantially. To be specific, the mAP@20 value of UDVH-TSN is 50.4% when using 128-bit code on FCVID, which is 4.8% higher than 45.6% achieved by UDVH-LSTM. That mainly owes to the effective learning of video-level representation via the combination of spatial and temporal CovNets provided by TSNs. When it comes to the most comparable competitor SSTH, the gap increases to 8.9%, which indicates the tremendous boost on the system performance achieved by UDVH-TSN. For those non-deep methods, satisfactory results still have been obtained because of the powerful feature modeling of the two-stream architecture adopted in the framework. In order to further validate the effectiveness of the proposed framework, we also plot Precision-Recall curves at 128 bits (Fig. 6(a)) and precision@100 (Fig. 7(a)) of all baselines separately. As can be seen, the best performance is still achieved by UDVH-TSN under those evaluation metrics, which is consistent with the results from mAP@K curves. For example, the precision@100 value of UDVH-TSN at 64 bits is 51%, which is at least about 3.6% higher than the most comparable baselines.

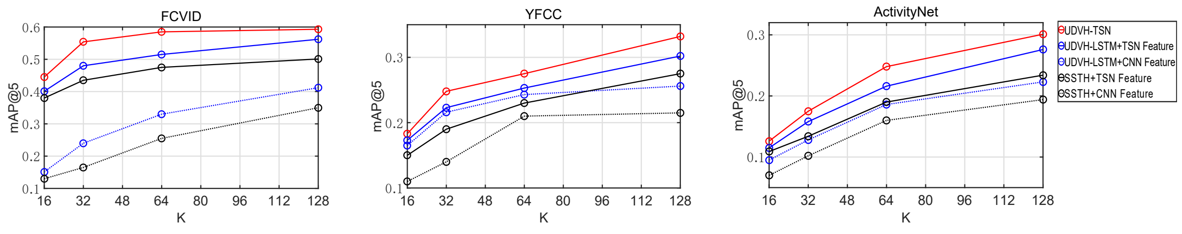
Experiment II on YFCC: Next, the experiments are conducted on YFCC dataset and the results are illustrated as following. Theoretically, the best performance is supposed to be given for all hashing methods on YFCC because it contains the most video clips among three datasets that can be utilized in the stage of unsupervised training. However, surprisingly, the retrieval performance drops down for all hashing methods compared to those on FCVID. Specifically for UDVH-TSN in Fig. 5(h), the difference between the mAP@20 values of UDVH-TSN (29.8%) and UDVH-LSTM (26.1%) at 128 bits is reduced to around 3.7%, while the value is 5.8% when compared with SSTH (24%). The possible explanation for this behaviour is the most videos in YFCC dataset are taken by mobile equipments from the amateurs rather than the professionals in FCVID [64]. The retrieval performance will be heavily affected when testing the hashing system with such low-quality video clips in YFCC. However, it is worth noting that UDVH-TSN still dominates those baselines with inspiring improvements, even dealing with the extremely challenging tasks. Similar to Experiment I, Precision-Recall (Fig. 6(b)) and precision@100 (Fig. 7(b)) curves on YFCC dataset are plotted as additional evaluations. Specifically, the precision@100 value at 64 bits when using UDVH-TSN is 30.2%, which is 3.9% higher than that under SSTH (26.3%). According to the figures, dominant performance still has been delivered by the proposed frameworks compared to other hashing methods on such challenging retrieval task, which consolidates the superiority of UDVH-TSN.

Experiment III on ActivityNet: Regarding the experimental results on ActivityNet dataset, the worst performance has been achieved by all hashing methods among three datasets comparatively. Particularly, the mAP@20 value of UDVH-TSN is 26.2% on ActivityNet as shown in Fig. 5(l), which is much lower than those on FCVID (50.4%) and YFCC (29.8%). A reasonable explanation for the unpleasant performance is that ActivityNet contains too many untrimmed videos with enormous intra-class variance, which lowers the hash code quality heavily and makes the retrieval more intractable [26], [48] compared to the experiments on FCVID and YFCC. When evaluating the performance on the same dataset under various baselines, UDVH-TSN still delivers the best performance according to the mAP@K figures as expected. Specifically, mAP@20 values for the two representative competitors: UDVH-LSTM and SSTH, are 20.7% and 17.8% respectively,



(a) FCVID (b) YFCC (c) ActivityNet

Fig. 8: The mAP@5 variations when using CNN and TSN features separately in the image-based hashing methods, where the solid and dot lines represent the results on TSN and CNN features, respectively.



(a) FCVID (b) YFCC (c) ActivityNet

Fig. 9: The mAP@5 variations when using CNN and TSN features separately in the video-based hashing methods, where the solid and dot lines represent the results on TSN and CNN features, respectively.

TABLE V: The training time at various code lengths on FCVID when using SSTH, UDVH-LSTM and UDVH-TSN. The time unit for network training is hour (h) and the rest processes are reported in second (s).

Method	**16 bits**			**32 bits**			**64 bits**			**128 bits**		
	FC	**BR**	**NT**	**FC**	**BR**	**NT**	**FC**	**BR**	**NT**	**FC**	**BR**	**NT**
SSTH [29]	/	/	8.21h	/	/	8.23h	/	/	8.48h	/	/	8.55h
UDVH-LSTM [19]	260.5s	0.38s	0.39h	263.2s	0.6s	0.39h	262.4s	1.1s	0.41h	265.1s	2.02s	0.41h
UDVH-TSN	267.3s	0.42s	0.74h	265.2s	0.62s	0.79h	270.2s	1.05s	0.81h	271s	2.1s	0.8h

which are at least 5.5% lower than 26.2% achieved by the proposed framework. Although the improvement is a bit limited, the retrieval performance by UDVH-TSN is still better than the most existing hashing methods. Surprisingly, even compared to some supervised methods like NSH [48] which utilizes ActivityNets as the benchmark, UDVH-TSN still gives promising results. For example, the mAP value at 64 bits of UDVH-TSN is 22.1%, which is 7.3% higher than 14.8% as reported in their paper of NSH. Followed by the same experiments on above datasets, other curves such as Precision-Recall (Fig. 6(c)) and precision@100 (Fig. 7(c)) are plotted separately to verify the superiority of the proposed framework. As can be seen from those figures, our framework consistently ranks top compared to other baselines. For example, the precision@100 at 64 bits reaches 26.1% under the proposed framework and the difference over SSTH (20.4%) is 5.7%. Those results show the superiority of UDVH-TSN clearly.

Above all, the proposed UDVH-TSN achieves excellent retrieval performance and outperforms the state-of-the-arts under extensive experiments on the testing datasets. In Fig. 10, the detailed retrieval results of top-5 returned videos when using 128 bits achieved by two comparable frameworks, UDVH-TSN and SSTH [29], are revealed. Due to the limited space, we randomly pick up some examples from three video datasets. Given six query videos of different topics, the proposed algorithm makes fewer mistakes in retrieving the similar videos for each query compared with SSTH. The major reason is that the neighborhood structure in the original data is well preserved in UDVH-TSN by incorporating the clustering into the unsupervised hashing process, thus producing more discriminative and effective binary codes for the nearest neighbour search tasks. It is also worth pointing out that DH [35], another deep-based hashing method in those baselines, yields unsatisfactory performance on all datasets, where three speculations are illustrated for those results as follows. First of all, it is essentially a deep method in image hashing, which cannot produce the video hash codes with temporal awareness [29]. Secondly, DH attempts to generate the binary codes via optimizing the approximate codes in the loss function, which lowers the hash code quality because of the huge gaps between the binary and real-valued spaces. The last one is that DH still fails to preserve the similarity structure from the original data in the code learning, which shares the similar drawback in SSTH.

6) Feature Selection: In this section, we make brief comparisons on the performance variations when using CNN and TSN features on three datasets, where the mAP@5 results at various code lengths for the image-based (e.g., ITQ, AGH, DH) and video-based hashing (SSTH, UDVH-LSTM and UDVH-TSN) methods are plotted in Fig. 8 and Fig. 9, respectively. Particularly, CNN features are extracted from the pre-trained VGG-19 [30] model and the corresponding results when adopting CNN features are directly reported from [19]. According to those figures, the great improvements have been achieved by using TSN features in both shallow and deep based baselines compared to those on CNN features, which mainly owe to the powerful modelling ability on temporal information from TSNs. Here, we focused on the video-based methods, including SSTH, UDVH-LSTM and UDVH-TSN, in Fig. 9, to estimate the feature impact on video hashing. Specifically, as shown in Fig. 9(a), the mAP@5 values at 128 bits on FCVID are 50.1% and 57.1% for SSTH and UDVH-LSTM, which are over 15% higher than the values achieved when using CNN features. For the other two datasets, the gaps are reduced to less than 10% because of the poor data quality as discussed in previous sections.

Moreover, the UDVH-based methods substantially yield better retrieval performance over SSTH, which is consistent with the results reported in Fig. 5 and further demonstrates the superiority of our self-taught training strategy involving clustering and balanced rotation. For examples, the mAP@5 values at the code length of 128 are 59.3% and 57.1% for UDVH-TSN and UDVH-LSTM on FCVID, which are at least 7% higher than that in SSTH (50.1%).

7) Efficiency Analysis: The efficiency issue is addressed in this part because it prevents such deep video hashing frameworks from being widely deployed in the real-world retrieval applications. As mentioned above, SSTH, UDVH-LSTM and UDVH-TSN are specifically designed for video hashing in those baselines, thus making the comparisons among them more persuasive. Consequently, the training expenses on FCVID dataset at various code lengths when using those methods are summarized in Table V. There are three sub processes during the training of UDVH-LSTM and UDVH-TSN: feature clustering (FC), balance rotation (BR) and network training (NT). For SSTH [29], we generally follow the settings in the paper and its released code[9] while adjusting network parameters accordingly and only the network training is required. The experiments are conducted on the same hardware configuration reported previously. As can be seen, the training time on FCVID dataset is around 0.5 and 0.9 hours per loop, which are calculated as the sum of the time values in those sub processes, when using UDVH-LSTM and UDVH-TSN individually. Considering the deep architectures usually converge within $t = 5$ loops for the UDVH-based methods, the total training cost is less than 4.5 hours, which is much shorter than training SSTH (over 8 hours), thus indicating the high efficiency of the proposed hashing framework. The most time-consuming parts of SSTH include the backpropagation of Binary LSTM units when updating their binary outputs and the complex network structures [29], which are not adopted in our algorithm. Moreover, it is worth noting that the proposed balanced rotation only costs a few seconds, which is highly competitive in the hashing applications.

[9]https://github.com/hanwangzhang/BLSTM

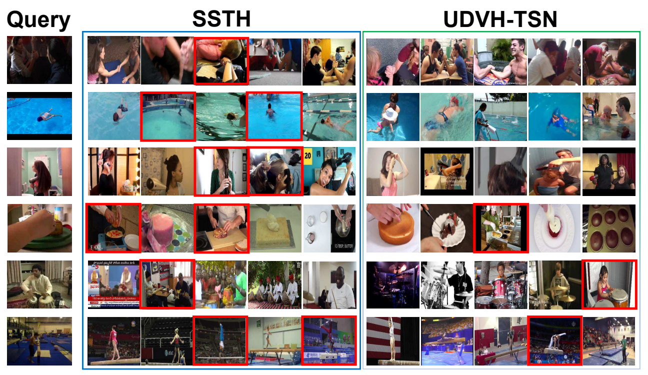


Fig. 10: Top-5 retrieval results when using SSTH and UDVH-TSN at the code length of 128 bits, where examples are randomly selected from three video datasets. The left column shows the query videos, the middle blocks and right blocks show the top-5 returned videos by SSTH and UDVH-TSN, respectively. Red rectangles indicate mistakes.

V. Conclusion

In this paper, we have presented a novel Unsupervised Deep Video Hashing for fast large-scale similarity search. In contrast to the previous video hashing approaches, feature learning and hash function learning are jointly integrated in a self-taught manner and optimized in alternating way within the deep architecture of UDVH. With the balance rotation applied in processing the video features, the variance of dimensions when projecting them into a low-dimensional space can be balanced, which improves the code quality. Our approach yields outstanding performance in the extensive experiments on three video datasets, which outperforms the state-of-the-arts significantly in terms of retrieval accuracy and efficiency. In future work, we will apply the proposed feature binarization to deal with other applications such as object detection [69], [70], object tracking [71], [72], and feature fusion [73], [74].

References

[1] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, “Learning to hash for indexing big dataa survey,” *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57, 2016.
[2] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 604–613.
[3] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, “Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.
[4] B. Xu, J. Bu, Y. Lin, C. Chen, X. He, and D. Cai, “Harmonious hashing.” in *IJCAI*, 2013, pp. 1820–1826.
[5] Z. Lin, G. Ding, J. Han, and J. Wang, “Cross-view retrieval via probability-based semantics-preserving hashing,” *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4342–4355, 2017.
[6] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, “Hashing with graphs,” in *ICML*, 2011, pp. 1–8.
[7] Y. Weiss, A. Torralba, and R. Fergus, “Spectral hashing,” in *NIPS*, 2009, pp. 1753–1760.

[8] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3424–3431.

[9] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search," in *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2010, pp. 18–25.

[10] Y. Guo, G. Ding, L. Liu, J. Han, and L. Shao, "Learning to hash with optimized anchor embedding for scalable retrieval," *IEEE Transactions on Image Processing*, vol. 26, no. 3, pp. 1344–1354, 2017.

[11] Y. Guo, G. Ding, and J. Han, "Robust quantization for general similarity search," *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 949–963, 2018.

[12] G. Ding, Y. Guo, J. Zhou, and Y. Gao, "Large-scale cross-modality search via collective matrix factorization hashing," *IEEE Transactions on Image Processing*, vol. 25, no. 11, pp. 5427–5440, 2016.

[13] W. Kong and W.-J. Li, "Double-bit quantization for hashing." in *AAAI*, vol. 1, no. 2, 2012, p. 5.

[14] J. Yue-Hei Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4694–4702.

[15] N. Srivastava, E. Mansimov, and R. Salakhutdinov, "Unsupervised learning of video representations using lstms." in *ICML*, 2015, pp. 843–852.

[16] C. Yan, H. Xie, D. Yang, J. Yin, Y. Zhang, and Q. Dai, "Supervised hash coding with deep neural network for environment perception of intelligent vehicles," *IEEE transactions on intelligent transportation systems*, vol. 19, no. 1, pp. 284–295, 2018.

[17] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] V. E. Liong, J. Lu, Y.-P. Tan, and J. Zhou, "Deep video hashing," *IEEE Transactions on Multimedia*, vol. 19, no. 6, pp. 1209–1219, 2017.

[19] G. Wu, L. Liu, Y. Guo, G. Ding, J. Han, J. Shen, and L. Shao, "Unsupervised deep video hashing with balanced rotation." IJCAI, 2017, pp. 3076–3082.

[20] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 20–36.

[21] Z. Lan, Y. Zhu, A. G. Hauptmann, and S. Newsam, "Deep local video feature for action recognition," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE, 2017, pp. 1219–1225.

[22] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, "Video summarization with long short-term memory," in *European conference on computer vision*. Springer, 2016, pp. 766–782.

[23] X. Wang, L. Gao, P. Wang, X. Sun, and X. Liu, "Two-stream 3-d convnet fusion for action recognition in videos with arbitrary size and length," *IEEE Transactions on Multimedia*, vol. 20, no. 3, pp. 634–644, 2018.

[24] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, 2015.

[25] Y. Peng, Y. Zhao, and J. Zhang, "Two-stream collaborative learning with spatial-temporal attention for video classification," *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[26] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 961–970.

[27] C. Yan, H. Xie, S. Liu, J. Yin, Y. Zhang, and Q. Dai, "Effective uyghur language text detection in complex background images for traffic prompt identification," *IEEE transactions on intelligent transportation systems*, vol. 19, no. 1, pp. 220–229, 2018.

[28] R. Veltkamp, H. Burkhardt, and H.-P. Kriegel, *State-of-the-art in content-based image and video retrieval*. Springer Science & Business Media, 2013, vol. 22.

[29] H. Zhang, M. Wang, R. Hong, and T.-S. Chua, "Play and rewind: Optimizing binary representations of videos by self-supervised temporal hashing," in *ACM Multimedia*. ACM, 2016, pp. 781–790.

[30] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[31] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," *arXiv preprint arXiv:1408.2927*, 2014.

[32] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2074–2081.

[33] C. Strecha, A. Bronstein, M. Bronstein, and P. Fua, "Ldahash: Improved matching with smaller descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 1, pp. 66–78, 2012.

[34] K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2938–2945.

[35] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *CVPR*, 2015, pp. 2475–2483.

[36] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3270–3278.

[37] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning." in *AAAI*, vol. 1, 2014, p. 2.

[38] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[40] J. Lu, V. E. Liong, and J. Zhou, "Deep hashing for scalable image search," *IEEE Transactions on Image Processing*, vol. 26, no. 5, pp. 2352–2367, 2017.

[41] Z. Chen, J. Lu, J. Feng, and J. Zhou, "Nonlinear discrete hashing," *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 123–135, 2017.

[42] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong, "Multiple feature hashing for real-time large scale near-duplicate video retrieval," in *ACM Multimedia*. ACM, 2011, pp. 423–432.

[43] L. Cao, Z. Li, Y. Mu, and S.-F. Chang, "Submodular video hashing: a unified framework towards video pooling and indexing," in *ACM Multimedia*. ACM, 2012, pp. 299–308.

[44] M. Douze, H. Jégou, and C. Schmid, "An image-based approach to video copy detection with spatio-temporal post-filtering," *IEEE Transactions on Multimedia*, vol. 12, no. 4, pp. 257–266, 2010.

[45] H. Wang and C. Schmid, "Action recognition with improved trajectories," in *ICCV*, 2013, pp. 3551–3558.

[46] G. Ye, D. Liu, J. Wang, and S.-F. Chang, "Large-scale video hashing via structure learning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2272–2279.

[47] M. Li and V. Monga, "Robust video hashing via multilinear subspace projections," *IEEE transactions on image processing*, vol. 21, no. 10, pp. 4397–4409, 2012.

[48] Z. Chen, J. Lu, J. Feng, and J. Zhou, "Nonlinear structural hashing for scalable video search," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 6, pp. 1421–1433, 2018.

[49] Y. Hao, T. Mu, R. Hong, M. Wang, N. An, and J. Y. Goulermas, "Stochastic multiview hashing for large-scale near-duplicate video retrieval," *IEEE Transactions on Multimedia*, vol. 19, no. 1, pp. 1–14, 2017.

[50] Y. Hao, T. Mu, J. Y. Goulermas, J. Jiang, R. Hong, and M. Wang, "Unsupervised t-distributed video hashing and its deep hashing extension," *IEEE Transactions on Image Processing*, vol. 26, no. 11, pp. 5531–5544, 2017.

[51] J. Song, H. Zhang, X. Li, L. Gao, M. Wang, and R. Hong, "Self-supervised video hashing with hierarchical binary auto-encoder," *IEEE Transactions on Image Processing*, vol. 27, no. 7, pp. 3210–3221, 2018.

[52] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[53] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[54] H. Zhang, L. Liu, Y. Long, and L. Shao, "Unsupervised deep hashing with pseudo labels for scalable image retrieval," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1626–1638, 2018.

[55] P. Zhang, W. Zhang, W.-J. Li, and M. Guo, "Supervised hashing with latent factor models," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 173–182.

[56] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3304–3311.

[57] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Fast supervised discrete hashing," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 2, pp. 490–496, 2018.

[58] 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,” *IEEE Transactions on Image Processing*, vol. 24, no. 12, pp. 4766–4779, 2015.
[59] Z. Wen and W. Yin, “A feasible method for optimization with orthogonality constraints,” *Mathematical Programming*, vol. 142, no. 1-2, pp. 397–434, 2013.
[60] J. Nocedal and S. J. Wright, *Sequential quadratic programming*. Springer, 2006.
[61] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.
[62] W. Kong and W.-J. Li, “Isotropic hashing,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1646–1654.
[63] P. Over, J. Fiscus, G. Sanders, D. Joy, M. Michel, G. Awad, A. Smeaton, W. Kraaij, and G. Quénot, “Trecvid 2014–an overview of the goals, tasks, data, evaluation mechanisms and metrics,” in *Proceedings of TRECVID*, 2014, p. 52.
[64] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li, “The new data and new challenges in multimedia research,” *arXiv preprint arXiv:1503.01817*, vol. 1, no. 8, 2015.
[65] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, “Sun database: Large-scale scene recognition from abbey to zoo,” in *CVPR*. IEEE, 2010, pp. 3485–3492.
[66] C. Zach, T. Pock, and H. Bischof, “A duality based approach for realtime tv-l 1 optical flow,” *Pattern Recognition*, pp. 214–223, 2007.
[67] K. Soomro, A. R. Zamir, and M. Shah, “Ucf101: A dataset of 101 human actions classes from videos in the wild,” *arXiv preprint arXiv:1212.0402*, 2012.
[68] J. Davis and M. Goadrich, “The relationship between precision-recall and roc curves,” in *ICML*. ACM, 2006, pp. 233–240.
[69] J. Han, D. Zhang, G. Cheng, N. Liu, and D. Xu, “Advanced deep-learning techniques for salient and category-specific object detection: A survey,” *IEEE Signal Processing Magazine*, vol. 35, no. 1, pp. 84–100, 2018.
[70] J. Han, R. Quan, D. Zhang, and F. Nie, “Robust object co-segmentation using background prior,” *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1639–1651, 2018.
[71] G. Ding, W. Chen, s. Zhao, J. Han, and Q. Liu, “Real-time scalable visual tracking via quadrangle kernelized correlation filters,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 140–150, 2018.
[72] J. Han, E. Pauwels, P. de Zeeuw, and P. de With, “Employing a rgb-d sensor for real-time tracking of humans across multiple re-entries in a smart environment.” *IEEE Transactions on Consumer Electronics*, vol. 58, no. 2, pp. 255–263, 2012.
[73] J. Han, E. Pauwels, and P. de Zeeuw, “Fast saliency-aware multi-modality image fusion,” *Neurocomputing*, vol. 111, pp. 70–80, 2013.
[74] Q. Zhang, Y. Liu, R. Blum, J. Han, and D. Tao, “Sparse representation based multi-sensor image fusion for multi-focus and multi-modality images: A review,” *Information Fusion*, vol. 40, pp. 57–75, 2018.

Gengshen Wu is currently a Ph. D. candidate with the School of Computing and Communications at Lancaster University, Lancaster, U.K.. His research interests include information retrieval, computer vision, and deep learning.

Jungong Han is a senior lecturer with the School of Computing and Communications at Lancaster University, Lancaster, U.K and was with the Department of Computer Science at Northumbria University, UK. Previously, he was a senior scientist (2012-2015) with Civolution Technology (a combining synergy of Philips CI and Thomson STS), a research staff (2010-2012) with the Centre for Mathematics and Computer Science, and a researcher (2005-2010) with the Technical University of Eindhoven in Netherlands.

Yuchen Guo received his B. Sc. degree from School of Software, and B. Ec. from School of Economics and Management, Tsinghua University, Beijing, China in 2013, and currently is a research fellow in School of Software in the same campus. His research interests include multimedia information retrieval, computer vision, and machine learning.

Li Liu received the Ph.D. degree from the Department of Electronic and Electrical Engineering, The University of Sheffield, Sheffield, U.K., in 2014. He is currently a research scientist at Inception Institute of Artificial Intelligence, Abu Dhabi, UAE. Previously, he was a research fellow with the School of Computing Sciences at the University of East Anglia, Norwich, UK.

Guiguang Ding received his Ph.D degree in electronic engineering from Xidian University, China, in 2004. He is currently an associate professor of School of Software, Tsinghua University. He has published 80 papers in major journals and conferences, including the IEEE TIP, TMM, TKDE, SIG IR, AAAI, ICML, IJCAI, CVPR, and ICCV. His current research centers on the area of multimedia information retrieval, computer vision and machine learning.

Qiang Ni is a Professor with the School of Computing and Communications, and with the Data Science Institute at Lancaster University, Lancaster, U.K. He received the B.Sc., M.Sc., and Ph.D. degrees in engineering from the Huazhong University of Science and Technology, China. His main research interests include the area of future generation communications and networking, including 5G and 6G, SDN, cloud networks, energy harvesting, IoTs, cyber physical systems, AI, machine learning, big data analytics, urban surveillance systems and smart city. He has authored or co-authored over 200 papers in these areas.

Ling Shao is the CEO and Chief Scientist of the Inception Institute of Artificial Intelligence (IIAI), Abu Dhabi, United Arab Emirates. His research interests include computer vision, machine learning and medical imaging. He is an associate editor of IEEE Transactions on Image Processing, IEEE Transactions on Neural Networks and Learning Systems, and several other journals. He is a fellow of the IAPR, the IET and the BCS.