Video Summarization: How to Use Deep-Learned Features Without a Large-Scale Dataset

Didik Purwanto, Yie-Tarng Chen, Wen-Hsien Fang and Wen-Chi Wu

August 24, 2018

Video Summarization: How to Use Deep-Learned Features Without a Large-Scale Dataset

Didik Purwanto, Yie-Tarng Chen, Wen-Hsien Fang, and Wen-Chi Wu
Department of Electronic and Computer Engineering
National Taiwan University of Science and Technology, Taipei, Taiwan, R.O.C.
Email: {d10602806,ytchen,whf,m10402151}@mail.ntust.edu.tw

***Abstract*—This paper proposes a framework incorporating deep-learned features with the conventional machine learning models within which the objective function is optimized by using quadratic programming or quasi-Newton methods instead of an end-to-end deep learning approach which uses variants of stochastic gradient descent algorithms. A temporal segmentation algorithm is first scrutinized by using a learning to rank scheme to detect the abrupt changes of frame appearances in a video sequence. Afterward, a peak-searching algorithm, statistics-sensitive non-linear iterative peak-clipping (SNIP), is employed to acquire the local maxima of the filtered video sequence after rank pooling, where each of the local maxima corresponds to a key frame in the video. Simulations show that the new approach outperforms the main state-of-the-art works on four public video datasets.**

***Index Terms*—Video summarization, key frame selection, temporal evolution, CNN, ranking machine.**

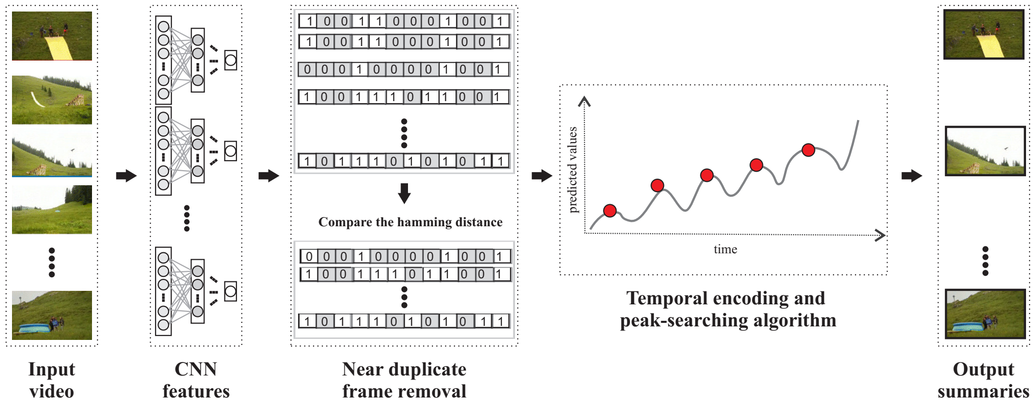
I. Introduction

With the proliferation of information and communication technologies such as wearable devices, self-centered cameras, video surveillance systems [1], the volume of video data has grown tremendously. As reported by YouTube [2], more than 300 hours duration of video are uploaded per minute to YouTube. Consequently, video summarization has become an extremely important video analysis tool and found applications in a wide range of applications like video retrieval [3], video indexing [4], and data-management [5]. However, video summarization is a challenging task, which needs to deal with some related intriguing issues, including diversity, the interestingness and the importance of videos. It is thus of great importance to develop an efficient and efficacious video summarization method to select the key frames to represent the videos.

A myriad of algorithms has been addressed for video summarization, including sequential determinantal point process [6], summary transfer [7], title video based summary [8], and online motion auto encoder [9]. Also, some unsupervised-based approaches that utilize a clustering algorithm, such as VSUMM [10], Delaunay Triagulation-based [11], and shot-boundary cluster [12] were investigated to group the frames into a set of clusters, the centroid of which was chosen as the key frame. Recently, with the success of Convolutional Neural Networks (CNN), Mahaseni *et al.* [13] used the generative adversarial network (GAN) which consisted of a summarizer and a discriminator to perform the unsupervised video summarization. Li *et al.* [14] proposed a model to summarize the videos based on a weighted combination of the importance, representativeness, diversity and smoothness of the storylinemeasures. Zhang *et al.* [15] estimated the range of inter-dependencies among frames using long-short term memory (LSTM) to conduct supervised learning for video summarization. However, training recurrent neural networks such as LSTM and GAN is not an easy task and a large annotated dataset is in general required to attain satisfactory performance. A question naturally arises: *Can we learn the temporal evolution of deep learned features for video summarization without a large scale of annotation data?*

To answer this question, in this paper, we propose a framework incorporating deep-learned features with the conventional machine learning models within which the objective function is optimized by using quadratic programming or quasi-Newton methods instead of an end-to-end deep learning approach which uses variants of stochastic gradient descent algorithms. As such, the proposed approach can work well even with a limited amount of training data. Since a shot boundary detection algorithm is a key component for video summarization, we, inspired by the success of rank pooling in modeling videos for action recognition, investigate a temporal segmentation algorithm by using a learning to rank scheme to detect the abrupt changes of frame appearances in a video sequence. To detect shot boundaries where the transition between shots is gradual, the near-duplicate frames are also detected and removed in advance by using an iterative quantization (ITQ) [16] with a locality preserved hashing function as a frame-based similarity measure. Although several deep hashing functions for visual similarity measures have been considered, these functions are always trained by approximate million images. In contrast, ITQ with deep-learned features can be directly trained by the existing benchmark datasets with limited annotation training samples as well. Also, a peak-searching algorithm, statistics-sensitive non-linear iterative peak-clipping (SNIP) [17], is employed to acquire the local maxima of the filtered video sequence after rank pooling, where each of the local maxima corresponds to a key frame in the video. Conducted simulations show that the new approach outperforms the main state-of-the-art works on four public

This work was supported by Ministry of Science and Technology of R.O.C. under the contracts MOST 106-2221-E-011-140 and MOST 107-2221-E-011-078-MY2.



Input video
CNN features
Compare the hamming distance
Near duplicate frame removal
predicted values
time
Temporal encoding and peak-searching algorithm
Output summaries

Fig. 1: Overview of the proposed video summarization.

video datasets.

In summary, the contributions of this paper include:

- A new framework for video summarization is devised based on modeling the temporal evolution of deep-learned features by rank pooling. The concept of learning to rank has been used for video summarization in a number of works [18, 19, 20]. The proposed approach, however, is entirely different from their methods in the following two aspects: First, we use the rank machine to enforce temporal evolution of CNN-learned frame features instead of scoring the importance of each frame trained based on the users preference. Second, the previous works require tedious annotation labels for scoring the importance of each frame. In contrast, we leverage the natural ordering of frames in a video to get rid of this.
- The removal of near-duplicate frames by ITQ and an effective key frame selection scheme based on SNIP are invoked to select a discriminative subset of key frames.

II. The Proposed Approach

In this section, we begin with the introduction of the overall pipeline in Sec. II-A. We then describe the iterative quantization to reduce the similar frames in Sec. II-B. Afterward, we explain the temporal encoding with rank pooling in Sec. II-C. Finally, we produce the final video summaries by utilizing the peak-searching algorithm, SNIP in Sec. II-D.

A. Overall Methodology

For easy illustration, the overall structure of the proposed method is as shown in Fig. 1, which consists of the following three main building blocks:

Iteration Quantization: Firstly, we use the generic CNN features as the input to ITQ, and a locality-preserved hashing function is used as a similarity measure for adjacent frames. If the hamming distance between the outputs of ITQ from two adjacent frames is less than a prescribed threshold, then one of them is regarded as redundant and removed from the video sequence.

Temporal Encoding: Secondly, the remaining CNN features are encoded by rank pooling to capture the temporal evolution of the frame-to-frame appearance.

Peak-Searching Algorithm: Finally, the predicted scores for the trimmed video sequence by rank pooling are regarded as a time series. SNIP is applied to search for the peak values and the corresponding frames are considered as key frames. The final video summary is obtained by summarizing these selected key frames.

B. Iteration Quantization

A video usually contains some similar frames, especially for the nearby or adjacent frames. These near-duplicate frame inevitably impact the performance of the video summaries. One approach to resolve this issue is to employ a preprocessing scheme to remove these near-duplicate frames. In this paper, we use ITQ [16] to remove the redundant frames by first hashing the CNN features into a set of binary codes for the purpose of efficiency. Afterward, ITQ computes the hamming distance between the two binary hashing codes corresponding to the two adjacent frames. If the distance is less than a prescribed threshold, the frame is considered as a near-duplicate frame and removed from the video.

To follow, we briefly describe how to use ITQ to learn the similarity between the adjacent frames. Suppose we have a set of n CNN features, $\{\mathbf{x}_i\}_{i=1}^{n}$ and are zero-centered, *i.e.* ,$\sum_{i=1}^{n}\mathbf{x}_i = \mathbf{0}$. Next, we learn a binary code matrix $\mathbf{B} \in \{-1,1\}^{n\times c}$, where c denotes the code length. Afterward, we perform PCA to reduce the dimension of the features. For a code of c bits, we obtain $\mathbf{W}$ by taking the c eigenvectors of the data covariance matrix $\mathbf{X}^T\mathbf{X}$, where $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$. If $\mathbf{W}$ is an optimal solution for PCA, so is $\tilde{\mathbf{W}} = \mathbf{WR}$ for any $c \times c$ orthogonal matrix $\mathbf{R}$. Based on a specific $\mathbf{B}$ and $\mathbf{R}$, the quantization error can be expressed as [16]

$$\begin{aligned}\mathbf{Q}(\mathbf{B},\mathbf{R}) &= \|\mathbf{B}-\mathbf{VR}\|_F^2 \\ &= \|\mathbf{B}\|_F^2 + \|\mathbf{V}\|_F^2 - 2tr\left(\mathbf{BR}^T\mathbf{V}^T\right) \qquad (1)\end{aligned}$$

where $\|\cdot\|_F$ denotes the Frobenius norm and ^{T} is the matrix transposition. The direct solution of (1) is a formidable task. Thereby, we resort to a suboptimal, yet more feasible k-means-like ITQ procedure to find a suboptimal solution. ITQ is comprised of two steps: First, we fix $\mathbf{R}$ and determine $\mathbf{B}$.

Next, we fix $\mathbf{B}$ and update $\mathbf{R}$. The overall procedure iterates between these two steps in a round-robin manner until

Algorithm 1 The peak searching algorithm

```
1: Input: A time series, $S(i)$ for $i = 1 \dots n$
2: Output: A clipping signal, $V(i)$ for $i = 1 \dots n$
3: Step 1. Initialization:
4: Set $m$ = 1 and the clipping signal, $V_1(i) = S(i)$ for $i = 1 \dots n$
5: Step 2. Compute new clipping signal $V_{m+1}(i)$ from the current clipping signal $V_m(i)$ and the clipping window, $V_m(i-m)$ and $V_m(i+m)$, using Eq. (6) for $i = m, m+1 \dots n-m$.
6: Set m = m + 1; go to Step 2 until $m$ = $M$
7: Step 3. $V(i) = V_M(i)$ for $i = 1 \dots n$
```

convergence. Also note that such an alternating minimization process is ensured to converge at least to a local optimal solution.

As the end of ITQ, we obtain a binary feature for every frame. Thereafter, we calculate the hamming distance between the two adjacent frames to detect their similarity. If the distance is less than a prescribed threshold, these frames are regarded as similar frames and removed from the video.

C. Temporal Encoding

After obtaining the filtered video sequence by ITQ, in this subsection, we attempt to capture the temporal evolution of the frame appearances with rank pooling [21] to summarize the videos. Given a set of m selected frames, say $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m\}$, we intend to model the chronological order of these selected feature vectors, *i.e.* $\mathbf{x}_m \succ \dots \succ \mathbf{x}_2 \dots \succ \mathbf{x}_1$, which can be derived as a pairwise ordering, $\mathbf{x}_{t+1} \succ \mathbf{x}_t$, *i.e.* if $\mathbf{x}_a \succ \mathbf{x}_b$ and $\mathbf{x}_b \succ \mathbf{x}_c \Rightarrow \mathbf{x}_a \succ \mathbf{x}_c$.

To this end, we can formulate this problem as a constrained minimization problem with pairwise ordering constraints. Specifically, we learn a linear function $\varphi(\mathbf{x}; \mathbf{u}) = \mathbf{u}^T \mathbf{x}$ with $\mathbf{u} \in R^D$ by pairwise linear rank pooling so as to satisfy the maximal margin constraint to avoid the over-fitting problems. Here, the score in rank pooling with respect to $\mathbf{x}_t$ can be computed by $\varphi(\mathbf{x}_t; \mathbf{u})$ and $\varphi(\mathbf{x}_{t+1}; \mathbf{u}) \succ \varphi(\mathbf{x}_t; \mathbf{u})$ to satisfy the pairwise constraints $\mathbf{x}_{t+1} \succ \mathbf{x}_t$. In other words, we need to learn a parametric vector $\mathbf{u}$ by solving the following constrained optimization problem [21]:

$$\begin{aligned} & arg \min_{\mathbf{u}} \frac{1}{2} \|\mathbf{u}\|^2 + C \sum_{\forall i,j \mathbf{x}_{t_i} \succ \mathbf{x}_{t_j}}^{1} \epsilon_{ij} \\ & s.t. \quad \mathbf{u}^T \left(\mathbf{x}_{t_i} - \mathbf{x}_{t_j}\right) \geq 1 - \epsilon_{ij}, \ \epsilon_{ij} \geq 0 \end{aligned} \tag{2}$$

where ϵ_{ij} represents a slack variable and C denotes a hyper-parameter, which can be determined by the cross-validation scheme.

This optimization problem can be readily solved by a linear rankSVM solver [22]. Afterward, the temporal evolution of a video sequence is encoded in the parameter vector $\mathbf{u}$. Alternatively, we can employ a vector-valued function, $\mathbf{v}_i$, of the CNN features of each frame as an input to rank pooling instead of the frame feature x_t. The vector-valued function can be classified as the independent frame representation and the moving average representation, which are defined respectively as follows.

• Independent frame representation: this function directly uses each independent frame as the output and is defined as

$$\mathbf{v}_t = \frac{\mathbf{x}_t}{\|\mathbf{x}_t\|} \tag{3}$$

• Moving average representation: this function aims at extracting the average behavior of the features within a temporal window with a fixed length to smooth the original features and is defined as

$$\mathbf{m}_t = \frac{1}{K} \sum_{\tau=t-K+1}^{t} \mathbf{x}_\tau \tag{4}$$

and

$$\mathbf{v}_t = \frac{\mathbf{m}_t}{\|\mathbf{m}_t\|} \tag{5}$$

where K is the length of the temporal window.

Based on the learned ranking function derived above, we obtain a time series with the frame number in the x-axis and the associated predicted score in the y-axis. This time series encodes the video-wise temporal evolution of frame appearances. We choose the local maxima of this time series as the key frames we are interested in. This is based on the observation that a local maximum of this time series indicates a dramatic change between the adjacent frames in its appearance, as it violates a pair-wise order constraint. Finally, a low-pass filter is employed to filter out the noise in the time series. As a result, we select those frames associated with the local maxima of the signal as key frames.

D. Peak-Searching Algorithm

Next, we describe an efficient peak-search algorithm, SNIP, [17] to find the local maxima of the time series. The rationale of SNIP is to iteratively invoke a clipping operation on a window, which is automatically adjusted to the width of the identified peak area, as summarized in Algorithm 1.

At the mth iteration, we first sequentially select a value of the clipping signal $V_m(i)$ with a sampling interval $[V_m(i-m), V_m(i+m)]$, called the clipping window. Thereafter, we compute the value of the new clipping signal $V_{m+1}(i)$ by using the smaller of $V_m(i)$ and the average of the values at two ends, *i.e.* $V_m(i-m)$ and $V_m(i+m)$ [17]. More specifically, $V_m(i)$ is updated by

$$V_{m+1}(i) = \min\left\{V_m(i), \frac{V_m(i-m) + V_m(i+m)}{2}\right\} \tag{6}$$

for $i = m \dots n-m$, where n is the number of points in the scoring signal. The new clipping signal, V_{m+1}, becomes smoother than V_m derived in the previous iteration, and the maximum number of iterations M can be determined by averaging key frames intervals from the training data or the key frame percentage from a video.

The main advantage of the SNIP algorithm is its capability to cope with the background shapes with a large variety. We can then determine the peak regions by overlapping the

TABLE I: Performance comparison of the proposed method with various setting.

Frame representation	OVP	YouTube	SumME	TVSum
Independent Frame				
256-dim	74.0%	62.3%	42.2%	56.3%
512-dim	77.6%	62.0%	42.8%	56.8%
1024-dim	75.9%	61.8%	42.5%	56.9%
Moving Average				
256-dim	74.2%	62.7%	42.8%	56.7%
512-dim	78.5%	63.1%	43.5%	57.4%
1024-dim	76.0%	62.4%	43.3%	57.0%

clipping signal produced by SNIP with the original time series, and the key frames will be selected based on this overlapping region.

III. Experimental Results

In this section, we first explain the details of the four video summarization datasets employed, and experimental setup in Sec. III-A. Sec. III-B assesses the performance of the proposed method, followed by a comparison with the state-of-the-art works through exhaustive computer simulations based on four public datasets in Sec. III-C.

A. Datasets and Experimental Setup

We conduct our experiments using four public video summarization datasets: Open Video Project [23, 10], YouTube datasets [10], SumMe datasets [24] and TVSum dataset [8]. Those datasets contain 25-50 videos with the length varying from 1 to 10 minutes. The simulations mainly follow the protocols and evaluation metrics provided in [11] in which the total key frames are less than 15% of the original frames. For each dataset [25, 10, 24, 8], we select 80 % of the videos for training and take the rest for testing. We run the experiments 100 times to obtain the final results. All of the simulations are implemented in the MATLAB environment on a computer with an Intel i7-6700 CPU, a 32 GB RAM, and a GTX 1080ti GPU. Finally, we use the Precision, Recall, and F-score to assess the performance of our work.

We use CNN fc-7 features as the input of our scheme. The RGB images are trained by using a pre-trained model by ImageNet and re-sized into 227 x 227 x 3. For temporal encoding, we use the moving average as an input scheme for the value vector function. The length of the binary code for the CNN feature space is set as 64 bits. For ITQ, the maximum number of iterations is set as 100.

B. Performance Evaluation

Assessment of Various Frame Representations: We compare the F-scores of our approach with a different frame representation of temporal encoding, as shown in Table I, from which we can see that encoding with the moving average can attain better performance than the independent frames on all four datasets because the independent frame representation cannot fully learn the temporal evolution of the video sequence. On the other hand, the moving average representation can produce smoother signals with less noise compared with the independent frame representation. Also, we investigate the performance in terms of F-score with various dimensions of ITQ features, as shown in Table I, from which we can see that 512-dim can achieve the best performance. This is perhaps due to the fact that smaller or higher dimensions cannot represent the features precisely - smaller dimensions may lose some important information while higher dimensions may contain more noise. Therefore, we use 512 dimensions in the remainder of this paper.

TABLE II: Performance comparison of the proposed work using rank pooling with and without ITQ and SNIP.

ITQ	SNIP	OVP	YouTube	SumMe	TVSum
		77.0%	60.5%	41.1	55.7
	√	77.5%	62.0%	41.8	56.3
√	√	78.5%	63.1%	43.1	56.9

Combination of Rank Pooling with and without ITQ and SNIP: We also inspect the performance of the proposed method with and without ITQ and SNIP to demonstrate the advantage of this combination. The comparison of the F-scores with a different combination is as shown in Table II, from which we can note that together with SNIP our approach which uses rank pooling can only obtain 0.5% to 1.5% performance improvement because SNIP can capture more distinct frame information. Next, with the additional incorporation of ITQ, the performance of the new approach can be further boosted by 0.6 % to 1.3 %. This is due to the fact that the removal of the redundant frames in the first stage can alleviate the false positive when performing video summarization.

For a vivid illustration, Figs. 2 - 4 show the selected key frames with and without ITQ and SNIP, from which we can see from these figures that together with ITQ and SNIP, the number of false negative indicated by black frames and that of false positive by the red cross is less than that without using these two schemes. As shown in Fig. 2, the results based on rank pooling only has a false positive, where a similar person with a different pose is selected. With ITQ in the preprocessing step, this frame can be removed.

C. Comparison with State-of-the-Art Methods

In this subsection, we compare the proposed approach with some recently reported works in terms of F-score on the four publicly available datasets. As a whole, eleven baselines are employed here for comparison, including: STIMO [26], VSUMM [10], SumTransfer [7], SUM-GAN [13], SeqDPP [6], LSTM [15], TVSum [8], Li *et al.* [14], MSDS-CC [27], LLR-SDS [28], and Online Motion AE [9]. The results from all of the baselines [26, 10, 6, 7, 13, 14, 15, 8, 27, 28, 9] are obtained from those reported in their papers.

For the OVP dataset, we compare our method with five baselines, STIMO [26], VSUMM [10], SumTransfer [7], SUM-GAN [13], and SeqDPP [6], in terms of the F-score, as shown in Table III, from which we can see that [10, 26] cannot provide satisfactory performance as they only use the handcrafted color features. SeqDPP [6] learned the natural

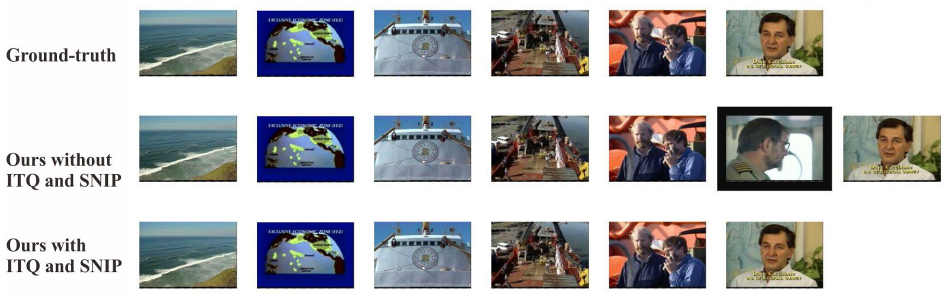


Fig. 2: Summarization results of Oceanfloor Legacy in the OVP dataset, where the black box represents the false positive.

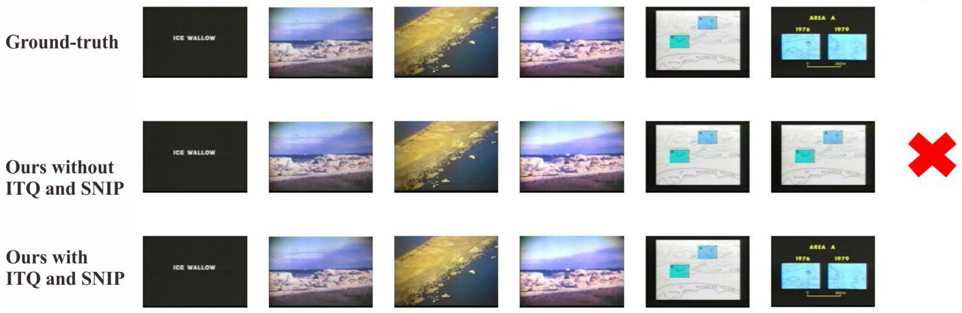


Fig. 3: Summarization results of Drift Ice as a Geologic Agent in the OVP dataset, where the red cross represents the false negative.

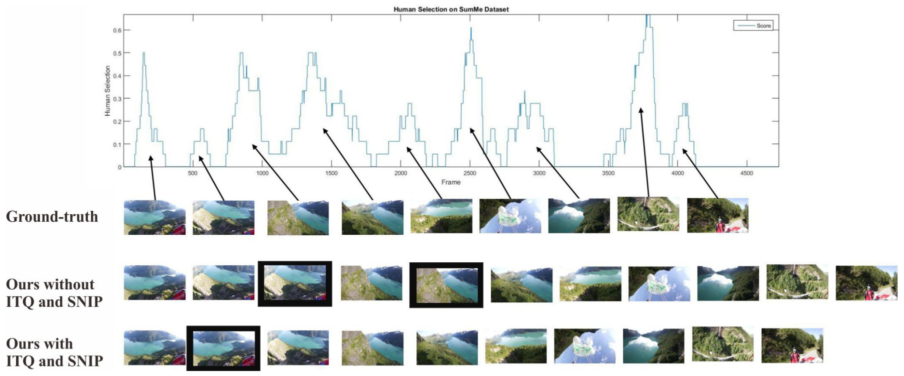


Fig. 4: Summarization results on Base jumping video in the SumMe dataset, where the black box represents the false positive.

TABLE III: Performance comparison with the state-of-the-art works on the OVP, YouTube, SumMe, and TVSum datasets, where the best results are bold-faced.

Methods	OVP	YouTube	SumMe	TVSum
STIMO [26]	63.4	-	-	-
VSUMM [10]	70.3	-	32.8	-
SumTransfer [7]	76.5	61.8	40.9	-
SUM-GAN [13]	77.3	62.5	41.7	56.3
SeqDPP [6]	77.7	58.7	-	-
LSTM [15]	-	-	41.8	54.7
TVSum [8]	-	-	-	51.3
Li *et al.* [14]	-	-	-	52.7
MSDS-CC [27]	-	-	40.6	52.3
LLR-SDS [28]	-	-	40.4	49.7
Online Motion AE [9]	-	-	37.7	51.5
Ours	**78.7**	**63.2**	**43.5**	**57.4**

order of the temporal structure in videos to obtain better performance. More recent approaches like SumTransfer [7] and SUM-GAN [13] are based on the CNN features and thus can attain superior performance compared with the aforementioned methods. Our approach, which uses rank pooling to learn the evolution of frame appearances with the removal of redundant frames beforehand, provides the best performance. Also, since this dataset consists of edited videos instead of raw videos, rank pooling can learn how the frame is evolving over the time and capture more significant changes at the moment of scene changes.

The comparison is also made on the YouTube dataset. Three baselines, SumTransfer [7], SUM-GAN [13], and SeqDPP [6], are compared, as shown in III, from which we can note that [13, 16], which treat the issue as a supervised learning problem, can attain satisfactory performance. SumTransfer [7] and SUM-GAN [13] outperform SeqDPP [6] as they use potent features extracted by the deep neural networks. Our method which uses rank pooling to learn the temporal evolution of the CNN features again can attain superior performance over all of the other works on this dataset.

For the SumMe dataset, we compare the proposed method with seven baselines, VSUMM [10], SumTransfer [7], SUM-GAN [13], LSTM [15], MSDS-CC [27], LLR-SDS [28], and

Online Motion AE [9] as shown in Table III, from which we can see that both VSUMM [10] and SumTransfer [7] cannot provide satisfactory performance, as employ the handcrafted features instead of more discriminative descriptors and thus cannot provide satisfactory performance. Also, we can find that our approach yields the best performance. It is because the SumMe dataset contain videos which have a slow changing shot scene. Thereby, ITQ, which can remove near-duplicate frames, is of importance to deal with this type of videos. Lastly, we compare the proposed method with seven baselines on the TVSum dataset, including, SUM-GAN [13], LSTM [15], TVSum [8], Li. *et al.* [14], MSDS-CC [27], LLR-SDS [28], and Online Motion AE [9] as shown in Table III. Our approach which combines the temporal encoding by rank pooling, near-duplicate frame removal by ITQ and peak-searching by SNIP can attain the best performance.

Note that in the above simulations, many works like SUM-GAN [13], Online Motion AE [9], LLR-SDS [28], MSDS-CC [27], Li *et al.* [14] and LSTM [15] are all based on the deep-learned features. They, however, require a large-scale annotated training videos to attain their best performance. On the other hand, the new approach combines the deep-learned features with the conventional rank pooling to circumvent this limitation and thus can outperform the aforementioned works on all four datasets.

IV. Conclusions

This paper has developed a CNN-based key-frame selection framework for video summarization. The new approach first removes redundant frames via ITQ and then performs rank pooling to acquire temporal evolution information. Finally, an iterative learning scheme, SNIP, is employed to capture the outliers which are regarded as key frames in the videos. By combining the deep-learned features with the conventional machine learning models, the new approach works well even with limited training data. Simulations show that the proposed method is superior to the state-of-the-art works on four public datasets.

References

[1] Kazuhito Takenaka, Takashi Bando, and Tadahiro Taniguchi. Automatic generation of summarized driving video with music and captions. In *Proceedings of Annual Conference of the IEEE Industrial Electronics Society*, pages 002409–002414, 2015.

[2] Youtube. Youtube statistics, 2018.

[3] Erkun Yang, Cheng Deng, Wei Liu, Xianglong Liu, Dacheng Tao, and Xinbo Gao. Pairwise relationship guided deep hashing for cross-modal retrieval. In *Proceeding of AAAI*, pages 1618–1625, 2017.

[4] Richang Hong, Lei Li, Junjie Cai, Dapeng Tao, Meng Wang, and Qi Tian. Coherent semantic-visual indexing for large-scale image retrieval in the cloud. In *IEEE Transactions on Image Processing*, volume 26, pages 4128–4138. IEEE, 2017.

[5] Yong Zhang, Meng Joo Er, and Mahardhika Pratama. Extractive document summarization based on convolutional neural networks. In *Proceedings of Annual Conference of the IEEE Industrial Electronics Society*, pages 918–922, 2016.

[6] Boqing Gong, Wei-Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. In *Proceeding of Advances in Neural Information Processing Systems*, pages 2069–2077, 2014.

[7] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. Summary transfer: Exemplar-based subset selection for video summarization. In *Proceeding of IEEE Conference on International Conference on Computer Vision*, pages 1059–1067, 2016.

[8] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. Tvsum: Summarizing web videos using titles. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5179–5187, 2015.

[9] Yujia Zhang, Xiaodan Liang, Dingwen Zhang, Min Tan, and Eric P Xing. Unsupervised object-level video summarization with online motion auto-encoder. In *arXiv preprint arXiv:1801.00543*, 2018.

[10] Sandra Eliza Fontes de Avila, Ana Paula Brandao Lopes, Antonio da Luz Jr, and Arnaldo de Albuquerque Araujo. VSUMM: a mechanism designed to produce static video summaries and a novel evaluation method. In *Pattern Recognition*, pages 56–68. Elsevier, 2011.

[11] Padmavathi Mundur, Yong Rao, and Yelena Yesha. Keyframe-based video summarization using Delaunay clustering. In *Proceeding of European Conference on Digital Libraries*, pages 219–232, 2006.

[12] I Cheng Chang and Kun-You Chen. Content-selection based video summarization. In *Proceedings of IEEE International Conference on Consumer Electronics*, pages 1–2, 2007.

[13] Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic. Unsupervised video summarization with adversarial LSTM networks. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, pages 2982–2991, 2017.

[14] Xuelong Li, Bin Zhao, and Xiaoqiang Lu. A general framework for edited video and raw video summarization. In *IEEE Transactions on Image Processing*, volume 26, pages 3652–3664. IEEE, 2017.

[15] Ke Zhang, Wei-Lun Chao, and Fei Sha Kristen Grauman. Video summarization with long short-term memory. In *Proceeding of European Conference on Computer Vision*, pages 766–782. Springer, 2016.

[16] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 35, pages 2916–2929. IEEE, 2013.

[17] Miroslav Morh'ac. An algorithm for determination of peak regions and baseline elimination in spectroscopic data. In *Nuclear Instruments and Methods in Physics Research*, pages 478–487. Elsevier, 2009.

[18] Min Sun, Ali Farhadi, Tseng-Hung Chen, and Steve Seitz. Ranking highlights in personal videos by analyzing edited videos. In *IEEE Transactions on Image Processing*, volume 25, pages 5145–5157. IEEE, 2016.

[19] Min Sun, Ali Farhadi, and Steve Seitz. Ranking domain-specific highlights by analyzing edited videos. In *Proceeding of European Conference on Computer Vision*, pages 787–802. Springer, 2014.

[20] Ting Yao, Tao Mei, and Yong Rui. Highlight detection with pairwise deep ranking for first-person video summarization. In *Proceedings of International Conference on Computer Vision*, 2016.

[21] Basura Fernando, Efstratios Gavves, Jose Oramas M., Amir Ghodrati, and Tinne Tuytelaars. Modeling video evolution for action recognition. In *Proceeding of IEEE Conference on Computer Vision and Pattern Recognition*, pages 5378–5387, 2015.

[22] Ching-Pei Lee and Chih-Jen Lin. Large-scale linear ranksvm. In *Neural computation*, volume 26, pages 781–817. MIT Press, 2014.

[23] Open video project. Open video project, 2018.

[24] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. Creating summaries from user videos. In *Proceeding of European Conference on Computer Vision*, pages 505–520. Springer, 2014.

[25] Wayne Wolf. Key frame selection by motion analysis. In *Proceeding of IEEE Conference on Acoustics, Speech, and Signal Processing*, pages 1228–1231, 1996.

[26] Marco Furini, Filippo Geraci, Manuela Montangero, and Marco Pellegrini. STIMO: STIll and MOving video storyboard for the web scenario. In *Proceeding of European Conference on Multimedia Tools and Applications*, pages 47–69, 2010.

[27] Jingjing Meng, Suchen Wang, Hongxing Wang, Junsong Yuan, and Yap-Peng Tan. Video summarization via multiview representative selection. In *IEEE Transactions on Image Processing*, volume 27, pages 2134–2145, 2018.

[28] Jingjing Meng, Hongxing Wang, Junsong Yuan, and Yap-Peng Tan. From keyframes to key objects: video summarization by representative object proposal selection. In *Proceeding of IEEE Conference on International Conference on Computer Vision*, pages 1039–1048, 2016.