VIDEO SUMMARIZATION WITH ANCHORS AND MULTI-HEAD ATTENTION

Yi-Lin Sung, Cheng-Yao Hong, Yen-Chi Hsu, Tyng-Luh Liu

Institute of Information Science, Academia Sinica

ABSTRACT

Video summarization is a challenging task that will automatically generate a representative and attractive highlight movie from the source video. Previous works explicitly exploit the hierarchical structure of video to train a summarizer. However, their method sometimes uses fixed-length segmentation, which breaks the video structure or requires additional training data to train the segmentation model. In this paper, we propose an Anchor-Based Attention RNN (ABA-RNN) for solving the video summarization problem. ABA-RNN provides two contributions. One is that we attain the frame-level and clip-level features by the anchor-based approach, and the model only needs one layer of RNN by introducing subtraction manner used in minus-LSTM. We also use multi-head attention to let the model select suitable lengths of segments. Another contribution is that we do not need any extra video preprocessing to determine shot boundaries and our architecture is end-to-end training. In experiments, we follow the standard datasets SumMe and TVSum and achieve competitive performance against the state-of-the-art results.

***Index Terms*—** Video summarization, multi-head attention, anchors, deep learning

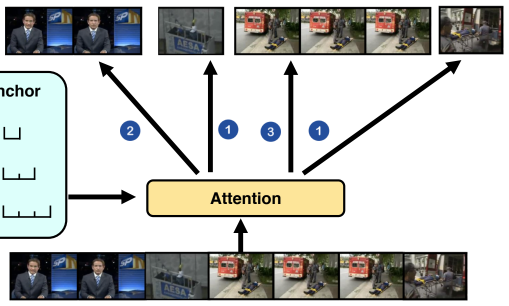

Fig. 1: Illustration of our anchors and attention mechanism. There are multiple anchors with different lengths, and the input frames use the attention mechanism to select the anchors with suitable lengths.

1. INTRODUCTION

In recent years, video data are increasing dramatically due to the popularity of video-sharing platforms and video-capturing equipment. Due to this reason, efficiently processing considerable daily videos becomes significant. Video summarization is one of the approaches to achieve this goal by retrieving the shorter edited video without losing important information.

Based on the continuous breakthroughs in deep learning research as well as the improvement of computational power, deep learning also has great success in video summarization. Zhang et al. use Bidirectional Long Short Term Memory (bi-LSTM), which can capture temporal information, to process the video sequences [1]. Mahasseni et al. combine Variational Autoencoders (VAE, [2]) and Generative Adversarial Networks (GAN, [3]) to learn robust summaries [4]. VAE learns to encode the video to a representative summary. GAN enhances the representativeness of the summary by introducing a discriminator, which tries to distinguish the whole video and the summary. [5] produces summaries that can maximize the designed diversity-representativeness reward by reinforcement learning.

It's well known that a video is composed of several shots[1], and a shot is composed of consecutive frames and [6, 7] leverage this hierarchical structure to enhance the Recurrent Neural Network (RNN) models. [6] groups the input to shots every k frames, and exploits hierarchical RNN as the summarizer. The first RNN is used to capture the local information within shots while the second RNN forms a global view by temporally processing the local information. The drawback of their approach is that the shot is generated by fixed-length segmentation, so they might not capture the accurate structure information [7] deals with this issue by using a sliding hierarchical RNN to both segments and summarize the video. By accurately segmenting the shot boundaries, the performance of their approach is better than previous work. However, one has to prepare additional data containing shot boundary labels, which are lacking in most video summarization dataset, to train the segmentation model.

Some existing works also introduce an attention mechanism to video summarization. The attention mechanism can assign importance weights to different shots/frames of the input instead of equally treating all the input ones. Therefore, it provides the inherent relations between the input video se-

[1]Shot/Clip/Segment is a sequence of images which contain similar contents, while information between shots/clips/segments should be distinct. We will alternatively use shot, clip and segment in this paper.

quence and the output. [8] build the attention from the decoder (summarizer) to the encoder and attained promising results. [9] leverages multi-head attention maps to enhance performance.

We were inspired by [10] to propose multiple sizes of anchors to bound the object. In this work, we form the various segmentation by multiple anchors with different lengths to solve fixed-length segmentation issues and the need for additional data. Moreover, unlike [10] using post-processing techniques to suppress non-optimal anchors, in our framework, each frame in the video will deploy multi-head attention mechanisms to attends the important anchors. The main idea is depicted in Fig. 1. Moreover, since the sequential nature in RNN, it suffers the speed issue when the input sequence is long. To ameliorate this, we apply the similar concept of LSTM-Minus [11] to extract the clip features. In this manner, we only need one layer RNN and the training and inference time is reduced.

2. OUR METHOD

2.1. Problem formulation

Given a video containing frames $X = (x_i)_1^n$ (n is the length of the video), the goal aims to predict the importance score of each frame, just like the demonstration in Fig. 2. In this stage, the model should give high scores for those representative and diverse frames. Afterward, several post-processing techniques are used, such as the knapsack algorithm, to form the final summaries whose lengths are less than 15% of which of original videos.

2.2. Hierarchical Gated Recurrent Unit

RNN can capture the temporal information from inputs, so it is often used in sequential inputs. In this work, we use Gated Recurrent Unit (GRU) [12] as our base RNN data since its good performance on a smaller dataset and memory efficiency.

First, we transform frames to deep features $\phi(x_i)$, $i = 1, ..., n$, by Convolutional Neural Networks (CNN), ϕ. Then we feed the transformed input to bidirectional GRU (bi-GRU) to extract the temporal information of forward and backward orders,

$$\begin{aligned} h_1^f, h_2^f, ..., h_n^f &= GRU^f(x_1, x_2, ..., x_n) \\ h_1^b, h_2^b, ..., h_n^b &= GRU^b(x_1, x_2, ..., x_n) \end{aligned} \tag{1}$$

And we denote $h_i = [(h_i^f)^T; (h_i^b)^T]^T$ as the i^{th} frame feature. After we have the forward and backward outputs after (1), we are going to construct the clip features by "feature subtraction" and "anchors".

2.3. Simulated clip features by subtraction and anchors

Using Kernel Temporal Segmentation (KTS) and other shot boundary detection methods such as [6, 7] have some shortcomings. The hyperparameters in KTS are hard to tune and are usually not the same across different videos. The other two methods also have their drawbacks, just like what we describe in the previous part. We use the difference of the hidden state in GRU to learn the hierarchical structure without explicit shot segmentation to simulate the clip features.

Inspired by [11], we apply the similar concept of LSTM-Minus to extract the frame features to obtain clip features and the illustration is shown in Fig. 2. For the forwarding pass of the GRU, let the hidden state of the timestep k_1 and k_2 ($k_2 > k_1$) are $h_{k_1}^f$ and $h_{k_2}^f$, we can use $h_{k_2}^f - h_{k_1}^f$ to represent the segment feature between the time step k_1 and k_2. The intuition behind this idea is that $h_{k_2}^f$ ($h_{k_1}^f$) is the accumulated embedding from time 1 to k_2 (k_1). Therefore, the residual between $h_{k_2}^f$ and $h_{k_1}^f$ are the features from time k_1 to k_2. Note that we can utilize similar idea to the backwarding pass, but in reversed order. For brevity, we let $Sub(d, k_1, k_2)$ to represent $h_{k_2}^d - h_{k_1}^d$, where $d \in \{b, f\}$ and $1 <= k_1 < k_2 <= n$. One can combine the forwarding and backwarding segment features as $BiSub(k_1, k_2) = [Sub(f, k_1, k_2); Sub(b, k_1, k_2)]$.

More specifically, we define the clip features with length $2l$ of i^{th} time step as

$$\begin{aligned} \bar{c}_i(l) &= BiSub(i - l, i + l) \\ \bar{C}_i &= \begin{bmatrix} | & | & & | \\ \bar{c}_i(l_1) & \bar{c}_i(l_2) & \cdots & \bar{c}_i(l_{N_L}) \\ | & | & & | \end{bmatrix} \end{aligned} \tag{2}$$

where $l \in S_L$ and it denotes the temporal range of the segment. $S_L = \{l_1, l_2, ..., l_{N_L}\}$ is the set containing N_L choices of the length of the segments, namely, the lengths of anchors. It means that we use the segment, whose center is at i^{th} frame, to be one of the clip features candidates for the i^{th} frame. $\bar{C}_i$ is the stacked clip features of all possible lengths, and it will be used in the next part. Then, we are going to construct the final clip feature of i^{th} frame c_i by all the candidates $\bar{C}_i$.

2.4. Clip-guided attention mechanism

An attention function can be described as mapping a query and a set of key-value pairs to an output, where the query, keys, values, and output are all vectors. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.

We apply the attention mechanism to combine the frame features and clip features. Attention is a function mapping a query(q) and a set of key-value (k and v) pairs to an output. A pair of similarities (distances) is first computed as the weight, and the output is the weighted sum of the values, namely,

$$\text{Attention}(q, K, V) = V^T \times \text{Softmax}\left(\frac{\text{Similarity}(q, K)}{\sqrt{d_q}}\right) \tag{3}$$

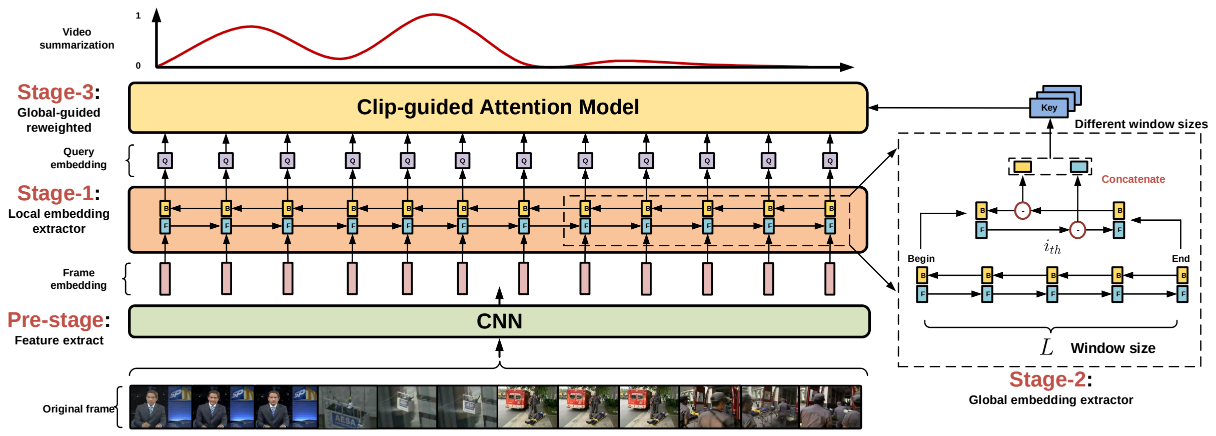
Video summarization
1
0
Stage-3: Global-guided reweighted
Clip-guided Attention Model
Query embedding
Stage-1: Local embedding extractor
Frame embedding
Pre-stage: Feature extract
CNN
Original frame
Key
Different window sizes
Concatenate
Begin
End
i_{th}
L Window size
Stage-2: Global embedding extractor

Fig. 2: Illustration of our model. Stage-1 is composed of bi-GRU to extract local information. Then, we exploit subtraction to obtained clip features with different lengths in Stage-2. In the end, each frame selects the clip features with suitable lengths by attention mechanism.

where q is vector and K, V are matrices, which forms by stacking all k and v, respectively, and the similarity function we use is the dot product between query and keys [13], that is $K \times q$. The scaling factor $\sqrt{d_q}$, where d_q is the dimension of q, is for preventing the large values of the dot product. Note that after q, k, v be feeding to the attention module, they are first transformed by additional linear projected layers. One can stack multiple attention modules by different linear projected layers to form so-called "multi-head attention" modules, and Fig. 3 depicts this idea.

In this work, we set queries are the frame features h_i, as keys and values are both stacked clip features of different lengths $\bar{C}_i$. We also employ a residual connection between the output and queries. Therefore, the output of our clip-guided attention module is $h_i+\text{Attention}(h_i, \bar{C}_i, \bar{C}_i)$. The "Similarity" module computes the weights between the frame and all the candidates while "Attention" outputs the weighted summation of $\bar{c}_i(l)$, and we treat it as the final clip features c_i. As the attention mechanism produces the clip features, the residual connection reserves the frame contents. And we feed the final output features to the linear classifier and sigmoid function to predict the probability of the frame being selected, namely,

$$p(h_i, \bar{C}_i) = \text{Sigmoid}(\text{linear}(h_i + \text{Attention}(h_i, \bar{C}_i, \bar{C}_i))) \tag{4}$$

2.5. Training

We train our model in an end-to-end manner. Let $(X, Y)^j$, is pairs frames-label data of j^{th} video, while $Y = (y_i)_1^{n_j}$ denotes whether the frame is keyframe or not. Therefore we can train our model as a sequential binary classification problem:

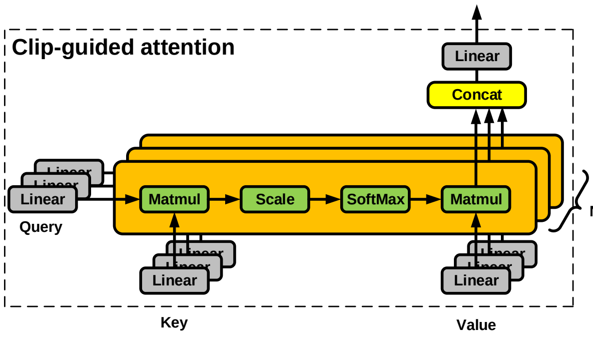

Fig. 3: Multi-attention module. It stacks N attention modules with different N linear projected layers.

$$\sum_{j=1}^{N} \frac{1}{n_j} \sum_{i=1}^{n_j} y_i \log(p(h_i, \bar{C}_i)) \tag{5}$$

where n_j is the length of the video, N is the total number of videos and $p(h_i, \bar{C}_i)$ is obtained through feeding X into (1), (2) and (4) sequentially. We name our whole model as Anchor-Based Attention RNN (ABA-RNN).

3. EXPERIMENTAL RESULTS

3.1. Dataset

Two benchmark datasets, SumMe [14] and TVSum [15] are mainly used to evaluate our approach. SumMe consists of 25 videos which cover holiday, event and sports. Their lengths range from about 1 to 6 minutes, and each video has 15 $\sim$ 18 frame-level importance scores. Besides, TVSum dataset contains 50 videos covering 10 different categories (5 videos per category). The duration of the videos is from 2 to 10 minutes. Apart from these two datasets, we also exploit Open Video Project (OVP) and Youtube [16], which has 50 and 39 videos, as auxiliary datasets to augment the training data.

3.2. Implementation Detail

For every frame of a video, we use the output of the penultimate layer of GoogLeNet [17] as its representation and use the representation as the input of GRU, whose dimension of the hidden state is set as 200. Training is stopped while it reaches 50 epochs, and the learning rate and the parameter of weight decay are 10^{-3} and 10^{-4}, respectively. The number of heads in the multi-head attention module sets to 3.

3.3. Evaluation

Let G be the generated machine summary, and U be the user summary. We compute precision (P) and recall (R) against the user summary, further exploiting F-measure (F) to evaluate generated summaries. P and R can be computed according to the temporal overlap between G and U, that is,

$$P = \frac{\text{the overlap of } G \text{ and } U}{\text{the length of } G}, \; R = \frac{\text{the overlap of } G \text{ and } U}{\text{the length of } U}$$

and the F-measure is the harmonic mean of P and R,

$$F = \frac{2 \times P \times R}{P + R} \times 100\%$$

Following previous works, we validate our method under three variants of settings: canonical (C), augmented (A), and transfer (T). In canonical setting, we use the 5-fold cross-validation on SumMe and TVSum datasets (80 % of videos for training and the rest for testing). In augmented settings, we still use the 5-fold validation, but we utilize additional training data with OVP and YouTube. In the transfer setting, we are going to test the transferability of our model. We choose one of the benchmark datasets (SumMe or TVSum) as a testing set and use the other three datasets to train the model.

Table 1: Ablation study on different lengths choices. The experiments are conducted in canonical setting.

	SumMe	TVSum
(1). $[1, 4, 16]$	47.9	60.0
(2). $[1, 8, 64]$	48.1	60.2
(3). $[4, 16, 64]$	**52.1**	**60.4**

3.4. Results

In this section, we are going to discuss how S_L influences our method. We explore the multiple lengths in S_L in SumMe and TVSum. Table 1 is the ablation results when the size of S_L is fixed to be three while the different lengths in S_L capture the different temporal relationships of a video. We figure out that $S_L = [4, 16, 64]$ got the best results for both datasets. Comparing (1) and (2), we can figure out that long term dependency is favorable for processing the videos between 1 and 10 minutes. We also observe that the smallest segment length can be larger from (1) and (2). Since the hidden states of two close frames are similar, the subtraction of them is nearly a zero embedding and becomes useless.

In Table 2, we show that ABA-RNN is comparable with other state-of-the-art methods. Note that ours does not use an additional complicate module such as VAE and GAN, and it might gain improvements by further integrating them into it. Table 2 also demonstrates the difference between our method with HRNN and HSA-RNN. The flexibility of anchors, ABA-RNN outperforms HRNN and is comparable with HSA-RNN, but without using additional shot-boundary labeled data. As illustrated in Fig. 4, ABA-RNN can precisely choose the key-frames in the video $\#12$ in TVSum, even when the target scores are rugged.

Table 2: Comparison with other methods on the SumMe and TvSum datasets in three experimental settings. * denotes that GAN is used to enhance the models.

	SumMe			TVSum		
Method	C	A	T	C	A	T
DPP-LSTM [1]	38.6	42.9	41.8	54.7	59.6	58.7
$\text{GAN}^*_{\text{sup}}$ [4]	41.7	43.6	-	56.3	61.2	-
$\text{DR-DSN}_{\text{sup}}$ [5]	42.1	43.9	42.6	58.1	59.8	58.9
HRNN [6]	-	41.1	-	-	57.7	-
HSA-RNN [18]	-	44.1	-	-	59.8	-
$\text{CSNet}_{\text{sup}}$ [19]	48.6	48.7	44.1	58.5	57.1	57.4
H-MAN* [9]	51.8	52.5	48.1	60.4	61.0	59.5
ABA-RNN	52.1	49.3	45.2	60.4	61.0	56.7

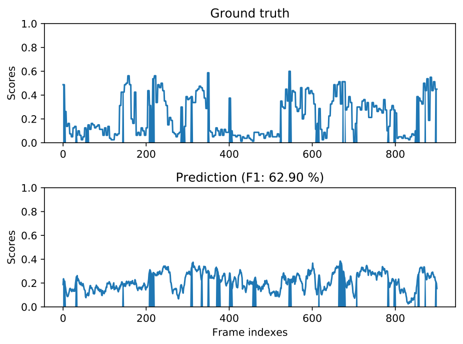

Fig. 4: Qualitative results of video $\#12$ in TVSum. The blue bars represent the key-frames.

4. CONCLUSIONS

In this paper, we proposed to use anchors to capture multiple lengths of segment features, and also exploit the multi-head attention mechanism to let each frame to select the segment feature with suitable lengths. The experiments demonstrate that our method is competitive with other methods.

Acknowledgement. This work was supported in part by the Ministry of Science and Technology (MOST), Taiwan under Grants 109-2634-F-001-012. We aregrateful to theNational Center for High-performance Com-putingfor providing computational resources and facilities.

5. REFERENCES

[1] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman, "Video summarization with long short-term memory," *CoRR*, vol. abs/1605.08110, 2016.

[2] Diederik P. Kingma and Max Welling, "Auto-encoding variational bayes," *CoRR*, vol. abs/1312.6114, 2013.

[3] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[4] Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic, "Unsupervised video summarization with adversarial lstm networks," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2982–2991, 2017.

[5] Kaiyang Zhou and Yu Qiao, "Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward," *ArXiv*, vol. abs/1801.00054, 2017.

[6] Bin Zhao, Xuelong Li, and Xiaoqiang Lu, "Hierarchical recurrent neural network for video summarization," in *Proceedings of the 25th ACM international conference on Multimedia*. ACM, 2017, pp. 863–871.

[7] Bin Zhao, Xuelong Li, and Xiaoqiang Lu, "Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7405–7414, 2018.

[8] Zhong Ji, Kailin Xiong, Yanwei Pang, and Xuelong Li, "Video summarization with attention-based encoder-decoder networks," *ArXiv*, vol. abs/1708.09545, 2017.

[9] Yen-Ting Liu, Yu-Jhe Li, Fu-En Yang, Shang-Fu Chen, and Yu-Chiang Frank Wang, "Learning hierarchical self-attention for video summarization," in *ICIP 2019*, 2019.

[10] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[11] Wenhui Wang and Baobao Chang, "Graph-based dependency parsing with bidirectional LSTM," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Berlin, Germany, Aug. 2016, pp. 2306–2315, Association for Computational Linguistics.

[12] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *SSST@EMNLP*, 2014.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[14] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool, "Creating summaries from user videos," in *ECCV*, 2014.

[15] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes, "Tvsum: Summarizing web videos using titles," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5179–5187, 2015.

[16] Sandra Eliza Fontes de Avila, Ana Paula Brandão Lopes, Antonio da Luz, and Arnaldo de Albuquerque Araújo, "Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method," *Pattern Recognition Letters*, vol. 32, pp. 56–68, 2011.

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[18] Bin Zhao, Xuelong Li, and Xiaoqiang Lu, "Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7405–7414.

[19] Yunjae Jung, Donghyeon Cho, Dahun Kim, Sanghyun Woo, and In So Kweon, "Discriminative feature learning for unsupervised video summarization," *CoRR*, vol. abs/1811.09791, 2018.