Video Summarization Guided by Local and Global Temporal Cues

Qinghao Ye
University of California, San Diego
q7ye@ucsd.edu

Liangde Li
University of California, San Diego
lil009@ucsd.edu

Chuanqi Yan
University of California, San Diego
c3yan@ucsd.edu

Weiqing Cao
University of California, San Diego
w5cao@ucsd.edu

Garrison W. Cottrell
University of California, San Diego
gary@eng.ucsd.edu

Abstract

Video summarization plays an increasingly important role in social media content filtering. In this paper, we use self-attention and graph convolutional neural networks to leverage local and global temporal cues. Our model integrates multi-scale temporal information and mines the relationship between video frames. We perform extensive experiments on SumMe and TVSum, and find that our method outperforms other state-of-the-art methods. Finally, we evaluate our approach qualitatively by visualizing the generated summaries.

1. Introduction

With the popularity of short video applications, a massive number of videos are uploaded by users around the world. YouTube users upload 1,000 hours of video every minute, making it impossible for anyone to catalog or index the content manually. Hence it is important to develop automatic methods for video retrieval and summarization. Video summarization segments the video into meaningful chunks, and then selects informative frames to summarize those chunks. This allows users to rapidly browse and navigate through a large video collection.

Video summarization has wide applications for users and content maintainers. For example, it allows video database maintainers to effectively index, browse, and promote their media assets. For video sharing platforms, such as TikTok, video summarization can improve the users' viewing experience. Content providers such as Netflix and HBO could use such a system to automatically extract the preview of the next episode of a series. Video summarization is crucial for these potential applications.

Motivation Current video summarization methods [15, 5] use LSTMs to process the videos and extract temporal features from them. However, these systems tend to ignore the global information in the videos. If segments of the video were permuted, many of these systems would likely generate similar results for the segments. Here, we incorporate multi-scale temporal information, from local segments to a global perspective in summarizing the video. We use graph convolutional neural networks to model the relationship between different temporal scale features with an adaptively learned graph. The architecture of our proposed model is illustrated in Figure 1.

The main contributions of our study can be highlighted as follows:

- We propose atrous temporal pyramid pooling and incorporate multi-scale temporal features into the modeling of global information, capturing information from multiple scales.
- We introduce a residual self-attention scheme which eases the vanishing gradient problem and leverages pairwise relations between frames.
- We combine an encoder-decoder based architecture and a graph convolutional neural network to model the local information for video frames.

- We performed extensive experiments on the SumMe and TVSum datasets and obtain state of the art results.
- Finally, we perform ablation experiments that demonstrate the effectiveness of the components of our model.

2. Related Work

Many video summarization technologies have been proposed in recent years. Zhang et al. [15] used Long Short-Term Memory (LSTM) for modeling the video sequence information and adopt Determinantal Point Process (DPP) which can help the model to find diverse sets of high-quality selection for generating a diverse video summary. Mahasseni et al. [8] use adversarial neural networks with LSTMs for generating the video summary, and the discriminator for selecting the summary. To complement the quality of generating the summary through adversarial LSTMs, Yuan et al. [14] adopt cycle consistency for external regularization. Zhao et al. [16] use a hierarchical RNN to model temporal information, while Ji et al. [5] add an attention mechanism on top of the RNN. Zhou et al. [17] introduce reinforcement learning and design the reward function for selecting the summary. Finally, Rochan et al. [10] adopt a fully convolutional neural network instead of using RNN to encode the temporal cue and predict the summary.

3. Proposed Method

The architecture of our proposed model is illustrated in the Figure 1. The basic idea of our method is that, in the global branch, we extract the temporal features of the input video with multiple dilated convolutional layers along the temporal dimension with different strides (a.k.a., Atrous Temporal Pyramid Pooling). Then we use the idea of self-attention proposed in [13] and residual learning introduced in [4] to learn a global representation from these multi-scale temporal features. In the local branch, we capture local temporal information and treat them as single node while regard the whole video as the graph, which indicates each clip has some relation with other frames (for example, a clip is prior of other clip). Therefore, we can use the Graph Convolution Network (GCN) [7] to the local representation of the input videos (a.k.a., local branch in the Figure 1). By incorporating and combining the local and global features, the model can output the probabilities of each frames to be selected as the keyframe.

3.1. Global Information Modelling

3.1.1 Atrous Temporal Pyramid Pooling

When we need to extract the features of a frame from its context, it is natural to combine the neighboring frames to build the time series. Also, the dilated convolution can consider the larger receptive field in an image under the same cost. In another words, the temporal dilated convolution enables model to capture the video with larger receptive field along the temporal dimension, leading to multi-scale video representation. Therefore, based on the intuition, we propose Atrous Temporal Pyramid Pooling (ATPP) based on dilated convolutions to extract video frame context time series features with multiple time scales. Suppose $w \in \mathbb{R}^{D\times w}$ is our dilated convolution kernel, the output of original feature input $F = \{f_1, f_2, f_3 \ldots, f_T\}$ through dilation rate r would be $F^{(r)}$:

$$F^{(r)} = \{f_1^{(r)}, f_2^{(r)}, f_3^{(r)} \ldots, f_T^{(r)}\}, \tag{1}$$

$$f_t^{(r)} = \sum_{i=1}^{w} f_{[t+r\times i]} \times w_{[i]}^{(r)}, \tag{2}$$

where $f_t^{(r)} \in \mathbb{R}^d$, and $w^{(r)}$ represent the dilated convolution kernel with dilation rate r. ATPP uses a set of r values to model multi-scale temporal information and processes the output of these dilated convolution in parallel. For example, ATPP includes N parallel dilated convolution with exponential dilation rates such as $r_n = 2^{n-1}$. Combining the the output from these N branches, we get new features F' as a series of f'_t:

$$f'_t = concat(f_t^{r_1}, f_t^{r_2}, f_t^{r_3} \ldots, f_T^{r_N}), \tag{3}$$

where $f'_t \in \mathbb{R}^{Nd}$ is the multi-scale temporal feature for time step t.

3.1.2 Residual Self-Attention

From the output of ATPP, we can get the multi-scale temporal features for each frame. To model the global information in the video, we introduce a residual self-attention mechanism to capture the relationships between the pairwise frames.

Self-attention is basically calculating a matrix M of size $T \times T$. For input $f'_t \in \mathbb{R}^{Nd}$, we use two separate fully connecting layers to embed the $F' = \{f'_t\}_{t=1}^{T}$ into k matrix $K \in \mathbb{R}^{(Nd/\alpha)\times T}$ and query matrix $Q \in \mathbb{R}^{(Nd/\alpha)\times T}$, and M is calculated as:

$$M = softmax(\frac{K^T Q}{\sqrt{ND/\alpha}}), \tag{4}$$

where M is a matrix including temporal and attention information of size $T \times T$. Then we also transform the input $f'_t \in \mathbb{R}^{Nd}$ into a value matrix $V \in \mathbb{R}^{(Nd/\alpha)\times T}$, and the final result of the residual self-attention is computed as:

$$F'' = VM + F', F'' \in \mathbb{R}^{Nd\times T}, \tag{5}$$

where α is the hyperparameter that controls the size of the model, and we empirically set $\alpha = 2$ for these experiments.

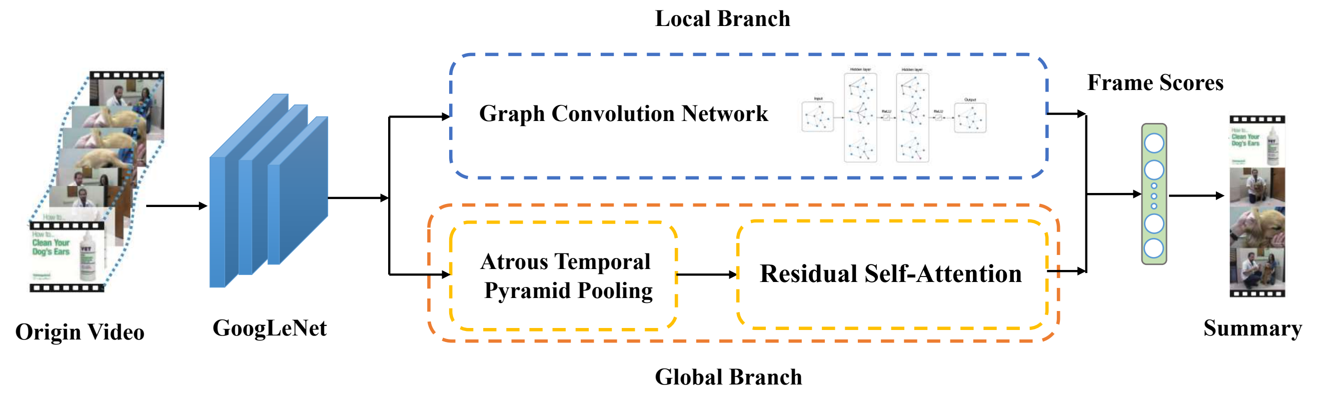


Figure 1. The overview of our proposed method. Our model uses two independent pathways, one for global information and the other for local feature extraction.

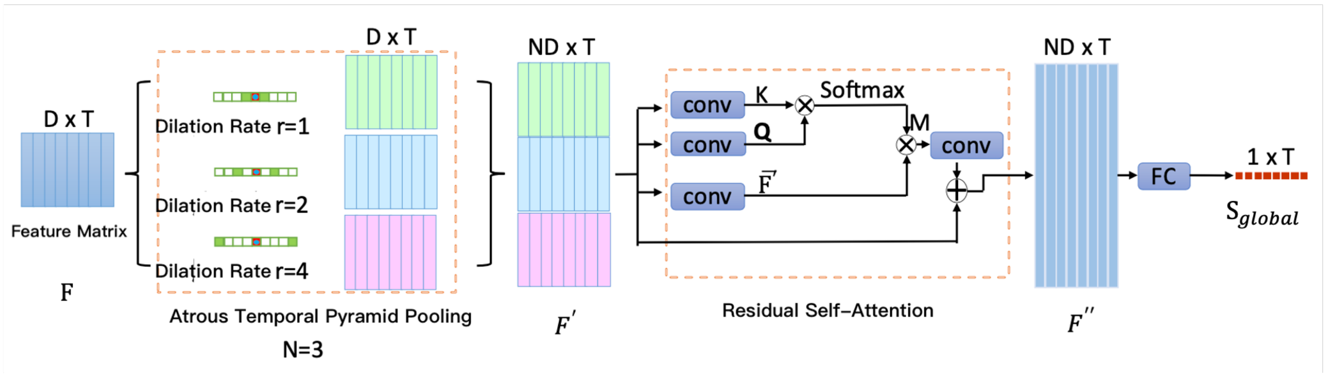


Figure 2. The workflow of the global information modeling scheme, with kernels' size 3.

Then, for every frame in F'' aggregated with attention and temporal information, we get its corresponding importance score by inputting it into a fully connected layer:

$$s_{global} = \sigma(F'' * W_{fc}), \tag{6}$$

where W_{fc} is a $Nd \times 1$ matrix of weight parameters of the network, $\sigma(\cdot)$ is the sigmoid function, and s_{global} is a vector of importance scores for all frames. The overall procedure of the global information modeling is illustrated in Figure 2

3.2. Local Information Modelling

In this subsection, we introduce the graph convolutional neural network for modeling the relation between adjacent frames and original frames, illustrated in Figure 3. The model has an encoder-decoder structure.

3.2.1 Encoder

The encoder performs down-sampling to get features in different temporal resolution. Down-sampling is achieved using 1D convolution with kernel size 3 and stride 2 along the temporal dimension, capturing the information in three neighboring frames, and results of this are convolved again with the same parameters, resulting in information from five neighboring frames. The output channel size is 1024. The original video frames after going through GoogLeNet, output the features $F = \{f_1, f_2, ..., f_T\}$, where T is the number of frames, and $f_i \in \mathbb{R}^{1024}$. Then, we feed them into two layers of 1D convolutions with weights w_1 and $w_2 \in \mathbb{R}^3$:

$$F_1 = F * w_1 = \{f'_1, f'_2, ..., f'_{T/2}\} \tag{7}$$
$$F_2 = F_1 * w_2 = \{f''_1, f''_2, ..., f''_{T/4}\} \tag{8}$$

where $*$ means convolution. Therefore, F_1 contains local information of 3 nearby video frames and F_2 contains information of 5 nearby video frames. We then perform convolution on the features F, F_1 and F_2 as shown below

$$F = F * w' \tag{9}$$
$$F_1 = F_1 * w'_1 \tag{10}$$
$$F_2 = F_2 * w'_2 \tag{11}$$

where w', w'_1 and w'_2 are convolution weights. Then, we concatenate the features to get F' and feed them into the graph convolution network:

$$F'' = f(F', A) = \sigma(AF'W) \tag{12}$$

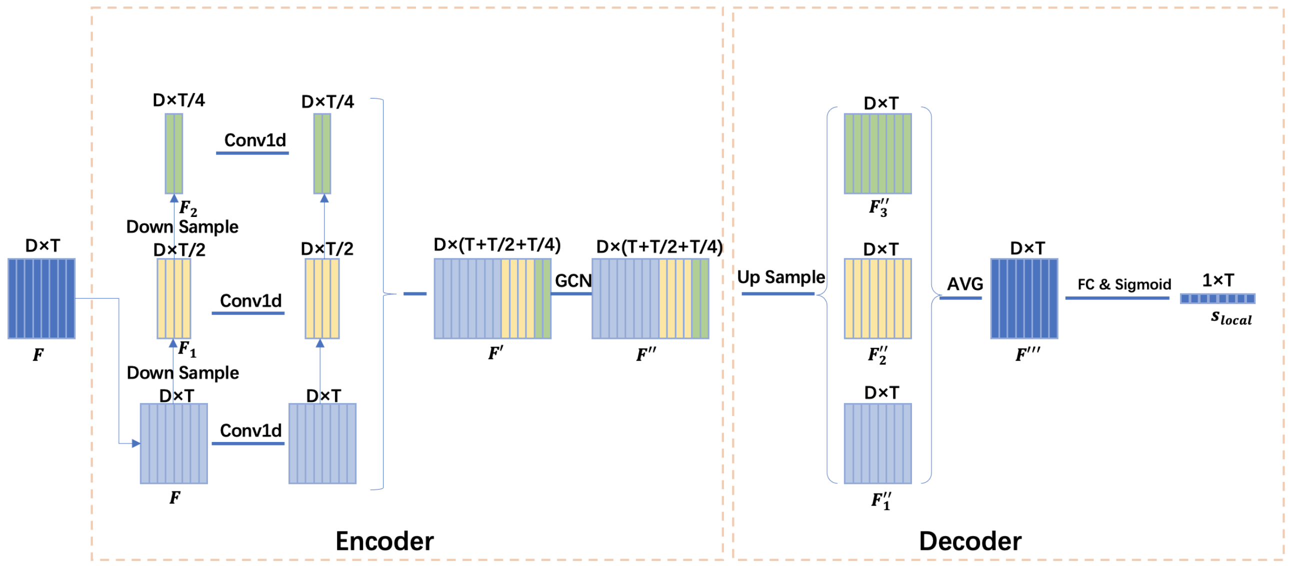


Figure 3. The workflow of the local information modeling scheme.

where W is initialized weights with shape $(T + \frac{T}{2} + \frac{T}{4}) \times (T + \frac{T}{2} + \frac{T}{4})$. More specifically, we use the original T frames, $T/2$ embedded features and $T/4$ embedded features as the nodes in the graph. A is an adjacency matrix trained based on F' which is computed as follows

$$A = softmax((F' * w_1'')^T (F' * w_2'')), \tag{13}$$

where w_1'' and w_2'' are two feature projection matrices for projecting the F' into two different feature spaces. As a result, the graph convolution output F'' contains both local and global information.

3.2.2 Decoder

The decoder part performs upsampling to the feature matrix of different time steps. When we were performing the down-sampling in the encoder part, we obtained a feature matrix of different time steps F, F_1 and F_2, which increased the receptive field of local features but decreased the size of the feature matrix. Then, we perform bilinear interpolation to recover these matrices to the original size. As a consequence, we calculate the mean of the three matrices and feed it into a fully connected layer to get the score of each frame.

In the upsampling part, we first divide the output feature matrix F'' from GCN according to the original size of F_1 and F_2, then we obtain F_1'', F_2'' and F_3'' using bilinear interpolation. After calculating their average, we obtain

$$F''' = \frac{1}{3}(F_1'' + F_2'' + F_3''), \tag{14}$$

where matrix $F''' \in \mathbb{R}^{D \times T}$. In order to get the local branch score for each frame, we transform it through a fully connected layer fc, which is computed as follows

$$s_{local} = \sigma(fc(F''')), \tag{15}$$

where $s_{local} \in \mathbb{R}^T$ are the local branch scores for each sample frame, and $\sigma(\cdot)$ is the sigmoid activation function.

3.3. Global and Local Score Fusion Scheme

In the previous sections, we derive global branch scores through the residual self-attention mechanism and local branch scores by the atrous temporal pyramid pooling and GCN.

Inspired by Ensemble Learning, we can aggregate multiple learners into a classifier with better performance when learners maintain their accuracy and there is still a certain degree of individual difference between them.

Based on that intuition, we firstly concatenate two score matrices into a vector $S' \in \mathbb{R}^{T \times 2}$, then a fully connected layer with 2 output units is applied and followed by the Softmax operation. Then we get the dynamic attention fusion weights $\alpha \in \mathbb{R}^{T \times 2}$ and compute the final score y_t for the t-th sample frame as:

$$y_t = \alpha_{t,1}(s_{global})_t + \alpha_{t,2}(s_{local})_t. \tag{16}$$

3.4. Summary Generation

For summary generation, we generate the final summary of an input video by selecting a set of key shots. Kernel Temporal Segmentation (KTS) [9] is first applied to generate a set of change points which indicates the start and end point of the key shot segments. Then, we constrain the summary length l to the proportion of user summary length to the original video length (usually 15% of the original video length). After probabilities of all frames in the input video have been obtained, we select key shots by the 0/1 Knapsack algorithm which is formulated as:

$$\max_{p_k} \sum_{k=1}^{K} p_k s_k, \quad s.t. \begin{cases} \sum\limits_{k=1}^{K} p_k l_k \leq l, \\ s_k = \frac{1}{l_k} \sum\limits_{t=1}^{l_k} y_k^t, \\ p_k \in \{0,1\}. \end{cases} \tag{17}$$

where s_k represents the mean score of a specific key shot within K key shots with length l_k, and the key shots with $p_k = 1$ are selected to generate the final summary.

4. Datasets and Metrics

We use four public benchmark datasets, SumMe [3], TVSum [11], YouTube [1], and OVP [1] for training and evaluation on SumMe and TVSum. SumMe consists 25 user-collected videos and corresponding annotations, and TVSum contains 50 user videos with 20 users' annotations for each video in frame-level. YouTube contains 39 videos and OVP has 50 videos. For fair comparison, we use the GoogLeNet [12] pretrained on ImageNet [2] to extract frame-level features for video frame representation. These datasets can be downloaded from their official websites.

For evaluation, we follow the protocol in [15, 14, 8], the similarity and quality of the generated summary is evaluated by measuring the agreement with user generated summary. We use the F_1 score to evaluate the video summary and 5-fold cross validation to evaluate the performance of the model.

5. Experiments

5.1. Evaluation Settings

From each frame of the video, we use GoogLeNet to extract features to obtain the feature vector of length 1024. We use binary cross-entropy as the loss function and Adam optimizer for training the model. We train our model for 100 epochs using PyTorch. During the experiment, we use grid search to tune the training and model hyperparameters.

We evaluate the model in three different settings. In the Canonical setting, both training and test set come from the same dataset. In particular, 80% of the dataset is used for training, and 20% of the dataset is used for testing. In the Augmented setting, we use 20% of the original dataset for testing and the training set contains 80% of the original dataset along with datasets from TVSum, OVP, and Youtube. In the Transfer learning setting, the training dataset only comes from TVSum, OVP, and YouTube and the test dataset comes from entirely SumMe. The detailed settings are shown in Table 1.

5.2. Quantitative Analysis

Experiment Details of Self-Attention Model The hyperparameters of the model include learning rate, dropout, and the number of convolution layers. The hyperparameter values searched over are listed in Table 2.

Experiment Details of CNN Based Encoder-Decoder Model The hyperparameters of the model include the learning rate lr, the weight_decay to control weight decay of learning rate, the dropout for fully connected layer, and most importantly the num_scale to control the number of times of encoder downsampling, i.e., the range of time scales. Table 3 presents the range of these hyperparameters. By analyzing the optimal set of hyperparameters in each data division, we conclude that a larger num_scale is needed when we have more datasets, because with large quantities and more categories of data, larger time scales are necessary to adapt to these complex datasets.

Comparison with State-of-the-art Models Bi-LSTM and DPP-LSTM [15] encode the time domain of the video with LSTMs. SUM-GAN [8] and its derivatives use generative adversarial networks to restructure the video frames with the generated video caption. DR-DSN [17] uses reinforcement learning to solve the video caption task. SUM-FCN [10] uses a deep fully convolutional neural network to encode the video. HSA-RNN [16] uses a hierarchical RNN to model the information of the video at different scales. CSNet [6] uses different step sizes and blocks to model the relationship between time domains of the video. The performance of each model on the SumMe dataset is shown in Table 4.

We can see that on the SumMe dataset, our SUM-Fusion method outperforms the current state-of-art on Canonical(CSNet) and Augmented(SUM-FCN) settings, by 5% and 4.2% respectively. And for the transfer setting, our model outperforms the current state-of-art, achieved by CSNet, by 2.9%. On the TVSum, our SUM-Fusion model achieves the new state-of-art on Canonical setting again, by outperforming CSNet 1.7%. It also has satisfying results on Augmented and Transfer settings. So generally, we achieves SOTA on Canonical setting on both datasets, as well as competitive performance on Augmented and Transfer.

In conclusion, when the distribution of the training set and test set do not differ, as in the Canonical setting, our fusion model outperforms the previous state of the art. But for the transfer dataset setting, since the training set and test set don't always come from the same dataset, the performance may not be better than a single model. The main challenge is improving the fusion mechanism to obtain better generalization, as the fused version does not always outperform the single models, and is sometimes worse than either alone. This suggests a conditional (e.g., mixture of experts) fusion

Setting	Train	Test
Canonical	80% SumMe	20% SumMe
Augmented	80% SumMe, augmented with data from TVSum, OVP and Youtube	20% SumMe
Transfer	TVSum, OVP and Youtube	SumMe

Table 1. The description of different evaluation settings used in our experiments.

Parameter	Values
lr	[1e-3,5e-4,1e-4,5e-5,1e-5,5e-6]
dropout	[0.5, 0.6, 0.7]
num_convs	[1,2,3,4]

Table 2. The search space of hyperparameters for the global information model branch.

Parameter	Values
lr	[5e-3,5e-4,1e-4,5e-5,1e-5,5e-6]
dropout	[1e-2,5e-3,1e-3,5e-4,1e-4,5e-5,1e-5]
weight_decay	[0.5, 0.7]
num_scale	[2,3,4]

Table 3. The search space of hyperparameters for the local information branch.

mechanism may improve our results.

5.3. Qualitative Analysis

We visualized the score of the original video from the standard dataset and the keyframes obtained from our model.

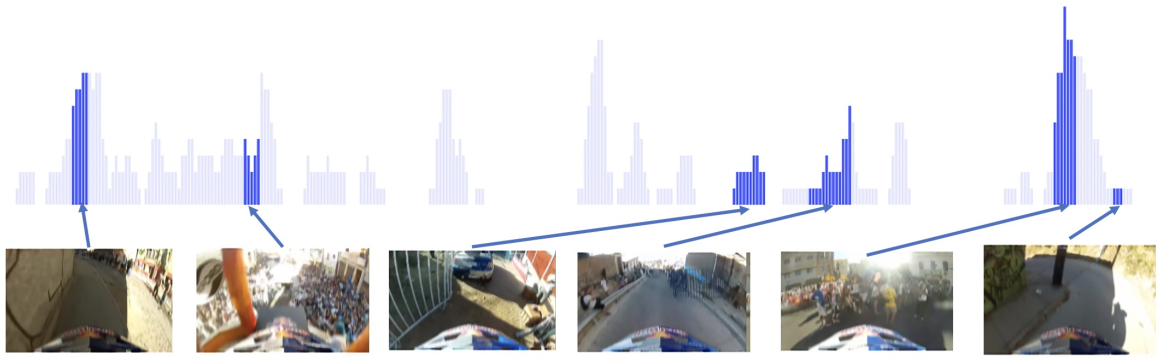
Figure 4 shows the result of our model based on self-attention captioning two videos. The bar on top of each figure shows the actual score of each frame, and the dark blue part is keyframes determined and obtained by our model. We can see from the figure that to a large extent, our model makes the right choices for the frames with higher scores.

6. Conclusion

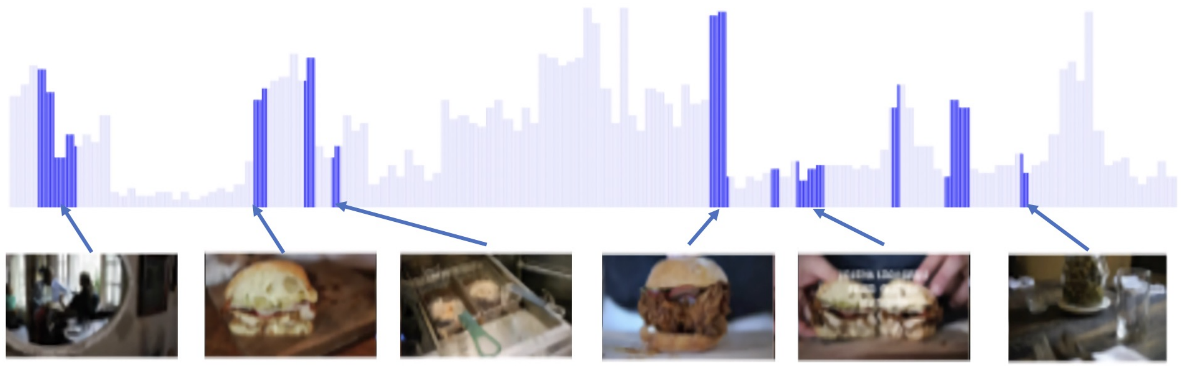
Video summarization is a challenging task, especially to pick the keyframes from a sequence of long videos. To capture features in different time scales into global information modeling. We used residual self-attention to ease the vanishing gradient problem in sequence modeling during training. To model frames locally in time, we used an encoder-decoder architecture with a graph neural network. We performed extensive experiments on the SumMe and TVSum dataset that demonstrated the effectiveness of our approach.

Method	Canonical (SumMe)	Augmented (SumMe)	Transfer (SumMe)	Canonical (TVSum)	Augmented (TVSum)	Transfer (TVSum)
Bi-LSTM [15]	37.6	41.6	40.7	54.2	57.9	56.9
DPP-LSTM [15]	38.6	42.9	41.8	54.7	59.6	58.7
SUM-GAN [13]	41.7	43.6	-	56.3	**61.2**	-
DR-DSN [17]	42.1	43.9	42.6	58.1	59.8	**58.9**
SUM-FCN [10]	47.5	51.1	44.1	56.8	59.2	58.2
HSA-RNN [16]	-	44.1	-	-	59.8	-
CSNet [6]	48.6	48.7	44.1	58.5	57.1	57.4
SUM-Global (ours)	51.1	51.3	**47.02**	58.2	58.2	58.5
SUM-Local (ours)	51.5	50.3	44.58	59.3	58.9	58.65
SUM-Fusion (ours)	**53.59**	**55.28**	43.45	**60.24**	59.22	58.2

Table 4. Performance comparison on SumMe and TVSum datasets measured by F_1-score.



(a) SumMe video 22:Valparaiso_Downhill



(b) TVSum video 20: Hamburger advertisements

Figure 4. Visualization of generated summaries for different videos. Light purple bars represent ground-truth scores, and dark purple bars denote generated summaries.

References

[1] Sandra Eliza Fontes De Avila, Ana Paula Brandao Lopes, Antonio da Luz Jr, and Arnaldo de Albuquerque Araújo. Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters*, 32(1):56–68, 2011.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009.

[3] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. Creating summaries from user videos. In *European conference on computer vision*, pages 505–520. Springer, 2014.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceed-*

ings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[5] Zhong Ji, Kailin Xiong, Yanwei Pang, and Xuelong Li. Video summarization with attention-based encoder–decoder networks. *IEEE Transactions on Circuits and Systems for Video Technology*, 30(6):1709–1717, 2019.

[6] Yunjae Jung, Donghyeon Cho, Dahun Kim, Sanghyun Woo, and In So Kweon. Discriminative feature learning for unsupervised video summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 8537–8544, 2019.

[7] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[8] Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic. Unsupervised video summarization with adversarial lstm networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 202–211, 2017.

[9] Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid. Category-specific video summarization. In *European conference on computer vision*, pages 540–555. Springer, 2014.

[10] Mrigank Rochan, Linwei Ye, and Yang Wang. Video summarization using fully convolutional sequence networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 347–363, 2018.

[11] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. Tvsum: Summarizing web videos using titles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5179–5187, 2015.

[12] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

[14] Li Yuan, Francis EH Tay, Ping Li, Li Zhou, and Jiashi Feng. Cycle-sum: cycle-consistent adversarial lstm networks for unsupervised video summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9143–9150, 2019.

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

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

[17] Kaiyang Zhou, Yu Qiao, and Tao Xiang. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.