SumGraph: Video Summarization via Recursive Graph Modeling

-Supplementary Materials-

Jungin Park[1⋆], Jiyoung Lee[1⋆], Ig-Jae Kim[2], and Kwanghoon Sohn[1⋆⋆]

[1] Yonsei University, Seoul, Korea
[2] Korea Institute of Science and Technology(KIST), Seoul, Korea
{newrun, easy00, khsohn}@yonsei.ac.kr
drjay@kist.re.kr

In this document, we provide more details of SumGraph and more experimental results on SumMe [2] and TVSum [7] dataset.

1 Network Architecture

Our networks are split into three parts: *initial graph construction network* to construct an initial graph using features from the pretrained feature extractor, *recursive graph refinement network* to obtain the initial graph by recursively estimating and refining the graph, and *summarization network* to perform the node classification. Although we use three graph convolution layers in the recursive graph refinement network, we can use fewer or more additional graph convolutional layers. The overall architectures of SumGraph corresponding to the number of graph convolution layers in the recursive graph refinement network are shown in Table 1.

2 Implementation and Training Details

Features. Following prior works [10, 11, 6], we uniformly sampled frames in the video to 2 fps. And then, we fed sampled frames into the pretrained GoogleNet [8] to extract the features from the *pool5* layer. Note that any feature representation(*e.g.*, C3D [9], I3D [1]) can be used for our model. For the fair comparison with previous works, we mainly use GoogleNet features.

Groundtruth. The datasets we used in experiments (*i.e.*, SumMe, TVSum, OVP, and YouTube) provide the groundtruth annotations in a different format. We record the groundtruth(GT) annotations used during training and testing for different datasets in Table 2. Following [10, 11, 6], we generate the single set of groundtruth keyframes for each training video from the multiple user annotations. For evaluation on testing videos, we convert the generated summaries and the groundtruth summaries to the interval-based keyshot summaries [11, 6]. While SumMe dataset provides groundtruth in the form of keyshots, TVSum dataset needs to be converted from keyframe annotations to keyshot-based

⋆ Both authors contributed equally to this work
⋆⋆ Corresponding author

1 Layer			
Layer	Ch I/O	Input	Output
G-conv1	1024/1024	$\mathbf{X}$	$\mathbf{Z}^k$
G-cls	1024/2	$\mathbf{Z}^k$	$\mathbf{Y}$
2 Layer			
Layer	Ch I/O	Input	Output
G-conv1	1024/1024	$\mathbf{X}$	feat1
G-conv2	1024/1024	feat1	$\mathbf{Z}^k$
G-cls	1024/2	$\mathbf{Z}^k$	$\mathbf{Y}$
3 Layer			
Layer	Ch I/O	Input	Output
G-conv1	1024/1024	$\mathbf{X}$	feat1
G-conv2	1024/2048	feat1	feat2
G-conv3	2048/1024	feat2	$\mathbf{Z}^k$
G-cls	1024/2	$\mathbf{Z}^k$	$\mathbf{Y}$
4 Layer			
Layer	Ch I/O	Input	Output
G-conv1	1024/1024	$\mathbf{X}$	feat1
G-conv2	1024/2048	feat1	feat2
G-conv3	2048/2048	feat2	feat3
G-conv4	2048/1024	feat3	$\mathbf{Z}^k$
G-cls	1024/2	$\mathbf{Z}^k$	$\mathbf{Y}$
5 Layer			
Layer	Ch I/O	Input	Output
G-conv1	1024/1024	$\mathbf{X}$	feat1
G-conv2	1024/2048	feat1	feat2
G-conv3	2048/4096	feat2	feat3
G-conv4	4096/2048	feat3	feat4
G-conv5	2048/1024	feat4	$\mathbf{Z}^k$
G-cls	1024/2	$\mathbf{Z}^k$	$\mathbf{Y}$

Table 1. Network configuration of SumGraph corresponding to the number of graph convolutional layers, where 'G-conv' and 'G-cls' denote graph convolutional layers in the recursive graph refinement network and the summarization network, respectively.

summaries. To generate keyshot-based summaries, we perform the procedure in [11] including the following steps: 1) apply KTS [5] to generate temporal segment in the forms of the disjoint intervals; 2) compute the average score for each segment and assign the score to frames in the interval; 3) apply the knapsack algorithm [7] to select frames so that the length of the keyshot groundtruth is below a certain threshold. In our experiments, the annotations are represented as a binary vector (0 for background and 1 for keyframe) with the number of elements is equal to the number of frames in a video.

Training and Optimization. While our SumGraph can be trained with the various lengths of videos, we uniformly sample frames from each video with a

Dataset	# annotations	Training GT	Testing GT
SumMe	15-18	frame-level scores	keyshots
TVSum	20	frame-level scores	frame-level scores
OVP	5	keyframes	-
YouTube	5	keyframes	-

Table 2. The format of groundtruth(GT) in training and testing phase for different datasets. In training, we convert from multiple user frame-level annotations to the single set of the keyframes for SumMe [2] and TVSum [7] datasets. In testing, we convert frame-level scores to the interval-based keyshot annotations for TVSum dataset following [11]. Note that OVP and YouTube datasets are not used in the testing phase.

# of iterations	0			1			2		
# of layers	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
1	20.1	25.3	22.4	33.3	34.3	33.8	40.1	40.8	40.4
2	21.3	27.7	24.1	33.6	33.5	33.6	41.5	42.5	42.0
3	24.7	30.1	27.1	34.3	35.5	34.9	41.0	44.8	42.8
4	23.2	27.8	25.3	35.1	37.6	36.3	43.6	44.6	44.1
5	22.6	25.0	23.7	33.5	37.1	35.2	42.5	42.5	42.5
# of iterations	3			4			5		
# of layers	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
1	45.1	49.5	47.2	48.8	49.9	49.3	48.4	50.0	49.2
2	46.6	49.9	48.2	50.3	51.0	50.7	51.2	52.9	52.1
3	50.6	52.3	51.4	50.9	52.0	51.4	52.2	52.9	**52.9**
4	49.2	53.2	51.1	**52.0**	52.3	52.1	50.9	**53.3**	52.1
5	49.5	51.8	50.6	50.3	52.1	51.2	51.2	52.5	51.9

Table 3. The quantitative results on SumMe [2] dataset in terms of precision, recall, and F-score corresponding to the number of graph convolution layers and the number of iterations.

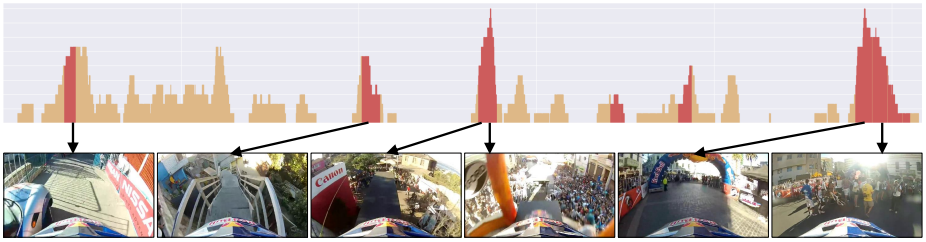
fixed temporal length of $T = 320$ for training to learn the parameters in the networks with mini-batch. During training, we set the batch size to 5 and learn all parameters of our model using Adam optimizer [3] with PyTorch [4]. The learning rate is set to 10^{-3} and decayed by a factor of 0.1 for every 20 epoch. We implement SumGraph on an Intel Core i7-7700 CPU with two NVIDIA Titan RTX GPU.

3 More Results

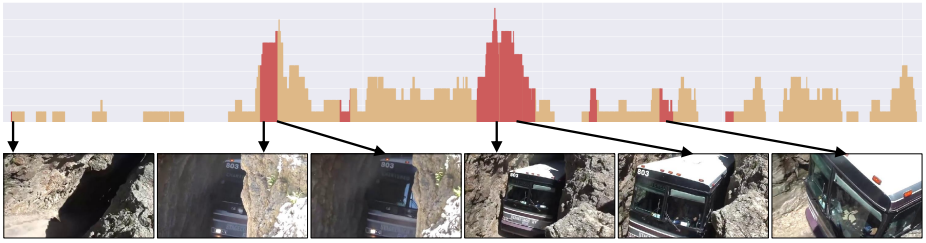
Qualitative Results. We show the more qualitative examples of selected keyframes using SumGraph on SumMe. We visualize the groundtruth importance scores and selected keyframes produced by SumGraph. For visualization of generated summaries, we sample 6 frames from the selected keyframes and display below the summary result, as shown in Fig. 1. In summary result, the marked red bars on brown backgrounds are the selected frames as summary.

# of iterations	0			1			2		
# of layers	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
1	30.1	31.7	30.9	44.1	39.9	41.9	54.3	52.1	53.2
2	32.9	32.0	32.4	43.3	42.6	43.0	59.7	55.5	57.5
3	36.5	32.4	34.3	46.2	40.1	42.9	57.1	56.4	56.7
4	35.8	32.5	34.1	46.8	44.9	45.8	59.7	55.6	57.6
5	33.6	31.1	32.3	44.3	44.0	44.2	55.4	55.1	55.2
# of iterations	3			4			5		
# of layers	Precision	Recall	F-score	Precision	Recall	F-score	Precision	Recall	F-score
1	58.4	55.7	57.0	60.1	58.8	59.4	60.3	59.1	59.7
2	62.1	61.1	61.6	64.5	62.4	63.4	60.8	60.4	60.6
3	64.3	63.5	63.9	64.3	64.0	64.1	64.5	**64.1**	**64.3**
4	63.4	63.2	63.3	63.5	61.9	62.7	64.6	61.5	63.0
5	63.3	61.7	62.5	63.4	63.2	63.3	**64.8**	63.2	64.0

Table 4. The quantitative results on TVSum [7] dataset in terms of precision, recall, and F-score corresponding to the number of graph convolution layers and the number of iterations.



(a)



(b)

Fig. 1. Qualitative results on the SumMe benchmark [2]: (a) Video number 22, (b) Video number 5. Brown bars show frame level user annotation. Red bars are selected as subset shots. Best viewed in color.

Ablation Study. We provide more quantitative results corresponding to the number of graph convolution layers in the recursive graph refinement network and the number of iterations on SumMe and TVSum datasets in Table 3 and Table 4. Note that the results with respect to iteration 0 represent the results

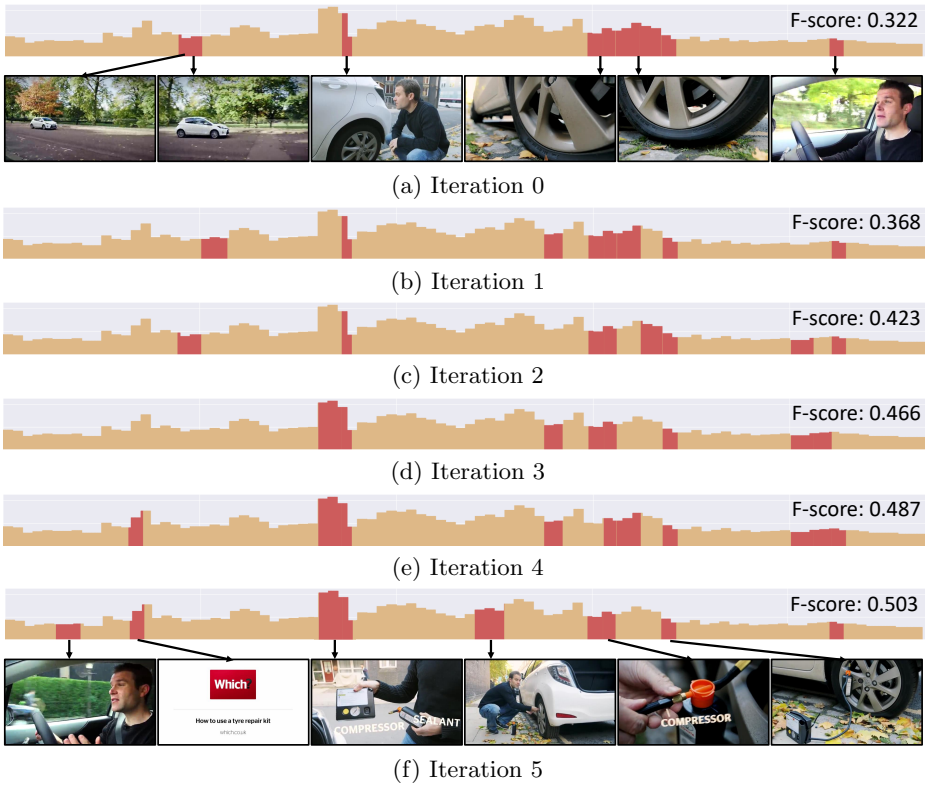


Fig. 2. Qualitative results for video number 2 on TVSum benchmark [7] according to the number of iteration. Brown bars show frame level user annotations, and red bars are selected as subset shots.

without any graph refinement. As mentioned in the main text, the summarization performance progressively converges as the number of iterations increases. The qualitative result for video number 2 on TVSum corresponding to the number of iterations is shown in Fig. 2. Brown bars show frame level user annotations, and red bars are selected subshots in every iteration. The result shows not only the summarization performance is improved, but also more informative parts of the video are selected as the number of iterations increased.

To investigate the efficiency of our model, we compare the number of parameters and the runtime with [6] in Table 5. As a baseline network, we used three graph convolution layers for the recursive graph refinement network and one graph convolution layer for the summarization network, and set the number of iterations for the graph refinement to five. For inference, while [6] takes about 84ms using about 116.5M network parameters for a 320 second video, our model takes 13ms using 5.5M network parameters under same settings. This

Method	# of parameters (M)	runtime (ms)
Rochan *et al.* [6]	116.5	84
Ours	5.5	13

Table 5. Comparison between [6] and our model in terms of the number of parameters and runtime.

# of iterations	0	1	2	3	4	5
runtime (ms)	4	6	7	9	11	13
F-score (%)	34.3	42.9	56.7	63.9	64.1	64.3

Table 6. Runtime and the summarization performance analysis for various numbers of iterations on TVSum [7] dataset.

result indicates that SumGraph surpasses [6] in terms of memory efficiency and summarization performance.

In addition, we evaluated the runtime and the summarization accuracy for different numbers of iterations. As shown in Table 6, our model with five iterations spent about three times of runtime more than the case of without any refinement, but achieved a performance improvement of 30%.

References

1. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: CVPR (2017)
2. Gygli, M., Grabner, H., Riemenschneider, H., Gool, L.V.: Creating summaries from user videos. In: ECCV (2014)
3. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
4. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
5. Potapov, D., Douze, M., Harchaoui, Z., Schmid, C.: Category-specific video summarization. In: ECCV (2014)
6. Rochan, M., Ye, L., Wang, Y.: Video summarization using fully convolutional sequence networks. In: ECCV (2018)
7. Song, Y., Vallmitjana, J., Stent, A., Jaimes, A.: Tvsum: Summarizing web videos using titles. In: CVPR (2015)
8. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: CVPR (2015)
9. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: ICCV (2015)
10. Zhang, K., Chao, W.L., Sha, F., Grauman, K.: Summary transfer: Examplar-based subset selection for video summarization. In: CVPR (2016)
11. Zhang, K., Chao, W.L., Sha, F., Grauman, K.: Video summarization with long short-term memory. In: ECCV (2016)