VideoSAGE: Video Summarization with Graph Representation Learning

Jose M. Rojas Chaves
Intel Corporation
jose.rojas.chaves@intel.com

Subarna Tripathi
Intel Labs
subarna.tripathi@intel.com

Abstract

We propose a graph-based representation learning framework for video summarization. First, we convert an input video to a graph where nodes correspond to each of the video frames. Then, we impose sparsity on the graph by connecting only those pairs of nodes that are within a specified temporal distance. We then formulate the video summarization task as a binary node classification problem, precisely classifying video frames whether they should belong to the output summary video. A graph constructed this way aims to capture long-range interactions among video frames, and the sparsity ensures the model trains without hitting the memory and compute bottleneck. Experiments on two datasets(SumMe and TVSum) demonstrate the effectiveness of the proposed nimble model compared to existing state-of-the-art summarization approaches while being one order of magnitude more efficient in compute time and memory.

1. Introduction

The landscape of video creation and consumption has been drastically changed in the last decade thanks to the affordable video capturing devices and the wide spread use of the Internet. Recent years have seen significant surge of social networks and video streaming, causing prevalence of user-created videos. Quick video browsing among massive video contents thus become essential. Video summarization is a way to facilitate quick grasping of video content by squeezing the most salient content from a long video to a short one.

To retain the most informative information of a video to its summarized version, we propose a framework leveraging short-range and long-range correlation among video segments. Our framework takes an input video and internally converts it to a graph. We impose sparsity constraints on the graph, so the internal representations remains manageable in a standard compute system, as well as retain its power of expressivity while learning from short-range and long-range correlations among different temporal segments.

To this end, we propose **VideoSAGE**, where each of the video frames become nodes of a graph and a subset of node pairs are connected to each other. We then learn parameters for graph convolutional networks on the internal graph representations and optimize our model for binary node classification signifying informative vs non-informative nodes for generating the video summary. Figure 1 demonstrates the framework pictorially. Please note that there is no ground truth graph. We can only compare our summarization model based on the end performance measured by summarization and correlation metrics, and can not objectively assess the graph construction otherwise.

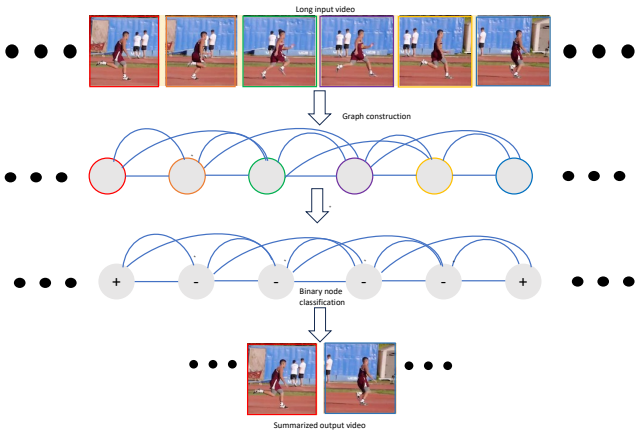

Figure 1. **VideoSAGE** constructs a graph from the input video, where each node corresponds to a video frame. Only those pairs of nodes are connected to each other who are within a temporal distance. Video summarization is thus formulated as a binary node classification problem for that graph. Our constructed graph has forward, backward and bidirectional edges. For visual clarity, we only show bi-directional edges in this figure. From top row to bottom row, the figure shows how regular input video is converted to a sparse graph, followed by binary node classification on nodes leading to summarized output video.

We perform experiments on two popular summarization datasets, namely TVSum and SumMe, and show the efficacy of our proposed method by the objective scores such as correlation metrics and F-1 scores. We also demonstrate

the qualitative results of the generated summarized videos. Most importantly, comparing with the existing state-of-the-art methods, **VideoSAGE** provides an order of magnitude faster inference time, smaller model size; 3× savings in peak memory footprint.

Our implementation is based on an open sourced library and the source code of our method is available on GitHub: https://github.com/IntelLabs/GraVi-T.

The novelty of our approach is in formulating the video summarization problem as a node classification on a graph. We construct the graph such that it enables interactions only between relevant nodes over time. The graph remains sparse enough such that the long-range context aggregation can be accommodated within a comparatively smaller memory and computation budget.

2. Related Work

Video summarization

Various machine-learning techniques have been employed for video summarization. These approaches can be categorized into two main groups: supervised [1, 3, 5, 7–9, 12, 21, 31, 32, 34, 35] and unsupervised [2, 10, 32, 33] methods. We primarily focus on the supervised learning methods in this section.

Numerous models in supervised learning have utilized annotated video datasets such as TVSum [25] and SumMe [14] for video summarization. One notable example of these models is A2Summ [7], which proposed a multimodal transformer-based model that aligns and attends multiple inputs (e.g., video, text, and sound) to select their most important parts. PGL-SUM [1] combined global and local multi-head attention mechanisms with a regressor network to select keyframes. CLIP-it [12] used a language-guided multi-head attention mechanism in addition to a multimodal transformer to generate summaries based on natural language queries. iPTNet [9] proposed an importance propagation based Teaching Network consisting of two separate collaborative modules that conducted video summarization and moment localization. In general, many other supervised learning methods rely also on self-attention mechanisms [3, 5, 8, 34] or LSTM networks [8, 31, 34] for frame importance prediction. Most state-of-the-art (SOTA) methods either use multimodality or a combination of multiple neural network techniques, making these methods relatively more complex. On the other hand, our proposed method achieves comparable results with a only fraction of memory and compute cost.

Graph-based video representation

While transformer models have recently took center stage in the research field of video understanding, in recent times, graph neural networks operated on explicit graph based representation are emerging for their complementary traits, including long-form reasoning thanks to the inherent low memory and compute requirements. Applications of specialized GNN based models in video understanding includes:visual relationship forecasting [13], dialog modelling [4], video retrieval [28], emotion recognition [24], action detection [30], video summarization [21], and others [15, 16, 19, 22].

Only a few methods utilize Graph Neural Networks (GNNs) for video summarization. RR-STG [35] built spatial and temporal graphs over which it performed relational reasoning with graph convolutional networks (GCNs) and extracted spatial-temporal representations for importance score prediction and key shot selection. RSGN [32] contained a summary generator and a video reconstructor. The first layer of the generator is a bidirectional LSTM that encodes the frame sequence in each shot, and the second layer is a graph model to explore the dependencies among different shots. And SumGraph [21]proposed a recursive graph modeling network consisting of 3 GCN layers plus another GCN working as a summarization layer. Unfortunately, no source code was found for any of these graph-based methods for results replication.

With only some exceptions [5, 21, 31, 32], most of these methods concentrated on keyframe predictions (including our proposal), and simply relied on using the knapsack algorithm, over predefined segments built with kernel temporal segmentation (KTS) and its respective predicted importance scores, for creating the final video summaries. The above context will be relevant while discussing the evaluation methods in section 4.2.

Our approach **VideoSAGE** is distinguished from the literature in the way of formulating the video summarization problem as a node classification on a graph constructed from the input videos. Our constructed graphs are sparse enough such that the long-range context aggregation can be accommodated within a comparatively smaller memory and computation budget, while capable of modeling short-range and long-range context aggregation.

3. Methodology

Figure 1 illustrates how **VideoSAGE** constructs a graph from an input video where each node corresponds to a frame of the video. This graph is able to reason over long temporal contexts over all nodes inspite of being not fully-connected. This is an important design choice to reduce memory and computation overheads [17, 18]. The edges in the graph are only between *relevant* nodes needed for message passing, leading to a sparse graph that can be accommodated within a small memory and computation budget.

We cast the video summarization problem as a node classification problem. To that end, we train a lightweight GNN to perform binary node classification on this graph. Inspired

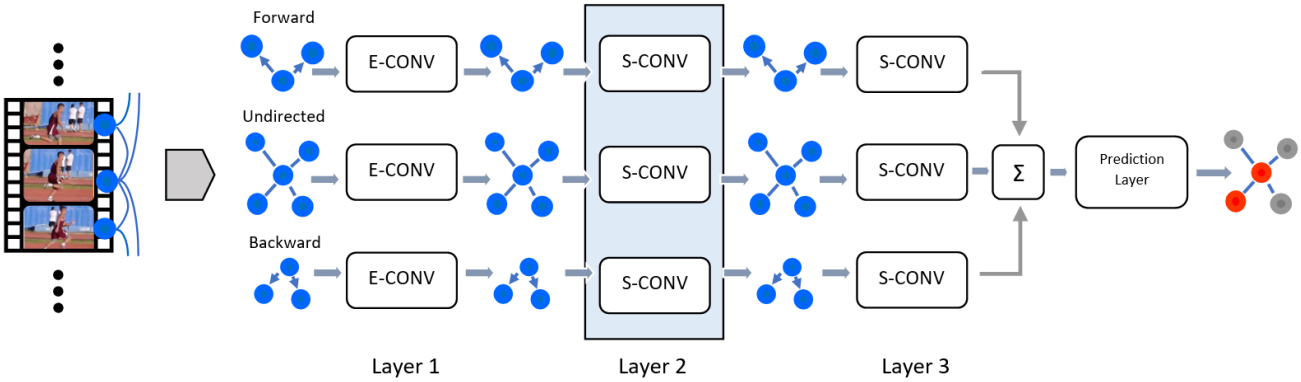

Figure 2. An illustration of utilized *Bi-directional* (a.k.a. *Bi-dir*) GNN model for video summarization. Here, we have three separate GNN modules for the forward, backward, and undirected graph, respectively. Each module has three layers where the weight of the second layer is shared by all three graph modules. The second layer is placed inside a solid-lined box to indicate the weight sharing while for the first and the third layer we use dotted-lines. E-CONV and S-CONV are shorthand for EDGECONV and SAGE-CONV, respectively.

by [18], our model utilizes three separate GNN modules for the forward, backward, and undirected graph, respectively. Each module has three layers where the weight of the second layer is shared across all the above three modules.

3.1. Notations

Let $G = (V, E)$ be a graph with the node set V and edge set E. For any $v \in V$, we define N_v to be the set of neighbors of v in G. We assume the graph has self-loops, i.e., $v \in N_v$. Let X denote the set of given node features $\{\mathbf{x}_v\}_{v \in V}$ where $\mathbf{x}_v \in \mathbb{R}^d$ is the feature vector associated with the node v. We can now define a k-layer GNN as a set of functions $\mathcal{F} = \{f_i\}_{i \in [k]}$ for $i \geq 1$ where each $f_i : V \to \mathbb{R}^m$ (m will depend on layer index i). All f_i is parameterized by a set of learnable parameters. Furthermore, $X_V^i = \{\mathbf{x}_v\}_{v \in V}$ is the set of features at layer i where $\mathbf{x}_v = f_i(v)$. Here, we assume that f_i has access to the graph G and the feature set from the last layer X_V^{i-1}.

- SAGE-CONV aggregation: This aggregation[6] is one of the widely used GCN type and has a computationally efficient form. Given a d-dimensional feature set X_V^{i-1}, the function $f_i : V \to \mathbb{R}^m$ is defined for $i \geq 1$ as follows:

$$f(v) = \sigma\Big(\sum_{w \in N_v} \mathsf{M}_i \mathbf{x}_w \Big)$$

  where $\mathbf{x}_w \in X_V^{i-1}$, $\mathsf{M}_i \in \mathbb{R}^{m \times d}$ is a learnable linear transformation, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function applied point-wise.
- EDGE-CONV aggregation: EDGE-CONV [29] models global and local-structures by applying channel-wise symmetric aggregation operations on the edge features associated with all the outgoing edges of each node. The aggregation function $f_i : V \to \mathbb{R}^m$ can be defined as:

$$f_i(v) = \sigma\Big(\sum_{w \in N_v} \mathsf{g}_i\big(\mathbf{x}_v \circ \mathbf{x}_w\big) \Big)$$

  where $\circ$ denotes concatenation and $\mathsf{g}_i : \mathbb{R}^{2d} \to \mathbb{R}^m$ is a learnable transformation. Usually, g_i is implemented by Multilayer perceptrons (MLPs). The number of parameters for EDGE-CONV is larger than SAGE-CONV. This gives the EDGE-CONV layer more expressive power at a cost of higher complexity. For our model, we set g_i to be an MLP with two layers of linear transformation and a non-linearity.

3.2. Video as a graph

We represent a video as a graph that is suitable for the task of summarization. In our implementation, the entire video is represented by a single graph *i.e.*if the video has n frames in it, the graph will have n nodes. We construct one graph for each video in the set.

The node set of $G = (V, E)$ is $V = [n]$, and for any $(i, j) \in [n] \times [n]$, we have $(i, j) \in E$ if the following two condition is satisfied: $|\mathsf{Time}(i)\text{-}\mathsf{Time}(j)| \leq T$ where Time is the time-stamp of each video frame and T is a hyperparameter for the maximum time difference between any connected node pairs. In other words, we connect two nodes (video frames) if they are temporally near by. Thus, the interactions between different video frames beyond just consecutive frames can be be modeled. To pose the video summarization task as a node classification problem, we also need to specify the feature vectors for each node $v \in V$. As done in several other SOTA methods such as [1, 14], we also use GoogLeNet [26] as the visual features for each video frame. By notation, a feature vector of node v is defined to be $x_v = [v_{\mathsf{visual}}]$ where v_{visual} is the

GoogLeNet feature of the video frame v. Finally, we can write $G = (V, E, X)$ where X is the set of the node features.

3.3. Video summarization as a node classification task

In the previous sub-section, we have described our graph construction procedure that converts a video into a graph $G = (V, E, X)$ where each node has its own visual feature vector. During the training process, we have access to the ground-truth labels of all video frames indicating whether they belong to the summarized output or not. Therefore, the task of video summarization can be naturally posed as a binary node classification problem in the above mentioned graph G, whether a node belongs to the output summary or not. Specifically, we train a three-layer GNN for this classification task (See Figure 2). The first layer in the network uses EDGE-CONV aggregation to learn pair-wise interactions between the nodes. For the last two layers, we observe that using SAGE-CONV aggregation provides better performance than EDGE-CONV, possibly due to EDGE-CONV's tendency to overfit.

Using the criterion: $|\mathsf{Time}(i) - \mathsf{Time}(j)| \leq T$ for connecting the nodes, the resultant graph renders undirected. In order to incorporate additional temporal ordering, we explicitly incorporate temporal direction as specified in [18]. The undirected GNN is augmented with two other parallel networks; one for going forward in time and another for going backward in time.

Specifically, in addition to the undirected graph, we create a forward graph where we connect (i, j) if and only if $0 \geq \mathsf{Time}(i) - \mathsf{Time}(j) \geq -T$. Similarly, (i, j) is connected in a backward graph if and only if $0 \leq \mathsf{Time}(i) - \mathsf{Time}(j) \leq T$. This gives us three separate graphs where each of the graphs can model different temporal relationships between the nodes. Additionally, the weights of the second layer of each graph is shared across the three graphs. This weight sharing technique can enforce the temporal consistencies among the different information modeled by the three graphs as well as reduce the number of parameters.

4. Experiments and Evaluation

4.1. Experimental set up

Datasets

We use two bench-marking datasets to evaluate the performance of our proposed **VideoSAGE** model. A copy of these datasets are downloaded from PGL-SUM's repository[1]. The SumMe [14] dataset is composed of 25 raw videos (1 to 6.5 minutes duration) covering holidays, events, and sports. And the TVSum [25] dataset consists of 50 YouTube videos (2 to 10 minutes duration) covering 10 categories selected from the TRECVid Multimedia Event Detection dataset [23]. Both datasets were originally evaluated by 15-20 different human users. And video summaries were built from each user evaluation by using the knapsack algorithm. Additionally, every video was down-sampled to 2 fps and provided with sampled-level importance scores (averaged from all users' inputs) and features (Size D = 1024) extracted from GooLeNet's [27] Pool 5 layer.

Experiments

To prepare the data inputs for our experiments, we create graph representations from each video in the datasets where each node corresponds to a frame within a temporal window of that video. This is done using **GraVi-T**[2] so the graph can reason over long temporal contexts for all nodes without being fully connected. Edges in the graphs are formed between temporally nearby nodes, up to 10 adjacent (backward and forward) frames for TVSum and 20 frames for SumMe, creating a sparse graph.

Each dataset was arranged into 5 splits following PGL-SUM's [1] approach. For each split, 20% of the videos were randomly selected for the validation set with the remaining 80% being used for training. Then, we train a lightweight GNN to perform binary node classification on each video graph, running 40 (SumMe) or 50 (TVSum) epochs over one split, the final model for one split is taken from the epoch with the lowest validation loss. Finally, evaluation metrics are gathered by running that model onto its validation set. This process is repeated over each split and the final evaluation results are the averages from all the splits.

To find a good set of hyper-parameters we follow an iterative process and utilize on RayTune's [11] grid search tool for exploration. The focus of the process is on finding stable learning curves and maximizing the Kendall's τ correlation result. The selected learning rate and weight decay values for SumMe were 0.001 and 0.003 respectively. For TVSum they were 0.002 and 0.0001 respectively. Batch size and dropout rate for both cases were 1 and 0.5 respectively.

4.2. Evaluation Metrics

Most existing SOTA methods have used the F1-Score and sometimes correlation metrics as well, to evaluate their models. The F1-Score is calculated as the maximum (SumMe) or the average (TVSum) value by comparing the predicted summary against each provided user summary from a given video, and then averaged from all videos in the validation set (as in [1, 7]). As pointed out in section 2, in most cases these summaries are created by running the Knapsack algorithm over predefined segments obtained

[1] https://github.com/e-apostolidis/PGL-SUM/tree/master/data

[2] https://github.com/IntelLabs/GraVi-T

Table 1. Comparison of results from uniformly randomly generated importance scores and results from perfect predictions (ground truth) using three evaluation methods on TVSum [25]: F1-Score, Otani et al [20], and our proposed method. * denotes reproduced results.

Predictions	F1-Score	Otani et al [20]*		Proposed Method	
	F1 (↑)	τ (↑)	ρ(↑)	τ(↑)	ρ(↑)
Random	54.37	0.00	0.00	−0.006	−0.009
Perfect	62.87	0.37	0.46	1.000	1.000

Table 2. Comparison with SOTA methods on the SumMe [14] and TVSum [25] datasets. We include the reproduced results of methods using GoogLeNet [26] features for a fair comparison. * denotes reproduced results.

Method	SumMe [14]			TVSum [25]		
	τ (↑)	ρ(↑)	F1(↑)	τ(↑)	ρ(↑)	F1(↑)
Random [20]	0.00	0.00	41.0	0.00	0.00	57.0
A2Summ [7]*	0.09	0.12	55.0	0.26	0.38	**63.4**
PGL-Sum [1]*	0.09	0.12	**55.6**	0.27	0.39	61.0
VideoSAGE (Ours)	**0.12**	**0.16**	46.0	**0.30**	**0.42**	58.2

through KTS. However, as stated by Otani *et al.* [20], the F1-Score result is mainly dictated by the video segmentation and its segment lengths, resulting into the contribution of the importance scores being completely ignored by the benchmark tests. This means that many SOTA works have been evaluating and comparing their models with a method that give more weight to that step they all have in common (Knapsack) and that can even ignore the piece they focus most and try to differentiate for the model.

Otani et al [20] proposes Spearman's ρ and Kendall's τ as correlation coefficients to evaluate models on how close the predicted scores are to human annotated scores. The correlation score for one video is then obtained by averaging the results over each individual annotations (20 user annotations on TVSum) and then the final score is obtained by averaging over the validation set.

An argument provided by Otani et al [20] against using the F1-Score method is that even using uniformly randomly generated predictions would result in a relatively high score (54.37%), comparable to other models at the time. On the other hand, calculating the respective correlation coefficients with this method would result in a near zero value, as one would have expected. See Table 1. However, if the model were perfect enough to predict its ground truth, Otani's method won't result in near perfect result ($\sim$1.0). This is because the result is the average over all user annotations, and it is impossible for the model to predict all of them at the same time. These annotations are subjective and they differ from user to user. In fact, the maximum value a model could technically achieve on TVSum is (62.87%).

We agree with Otani et al. about the F1-Score and provide its result values here only for reference. We choose a more traditional way of evaluating our model, that is comparing against the ground truth importance scores. Which we consider as a fairer approach since it was what the model was trained to predict. A perfect prediction with this method would give a perfect result (1.0), as seen in Table 1. Providing a complete [-1,1] range for evaluation and comparison.

4.3. Results

We compare the proposed method **VideoSAGE** with the previous SOTA methods on SumMe [14] and TVSum [25] datasets as shown in Table 2. A2Summ [7] and PGL-Sum [1] are two of the best methods bench-marked on the above two datasets. For a fair comparison, we have replicated results from these two models to calculate correlation coefficients by following the method proposed in section 4.2.

Results from table 2 show that even when A2Summ [7] and PGL-Sum [1] have better F1-Scores, it is **VideoSAGE** which predicts the importance scores better. **VideoSAGE** beats both the above models, on both datasets, on their Kendall's τ and Spearman's ρ correlation coefficients by 3-4%, showing its superiority. In the following section (4.4), we will exemplify why having better F1-Score does not always mean better prediction or better model.

4.4. Qualitative Results

Two videos were selected to showcase the qualitative results: Video_16 from TVSum [25] and video_1 from SumMe [14]. Video_16 was selected as its individual correlation results (Table 3) where near to the general average (Table 2) on **VideoSAGE**. On the other hand, Video_1 is an outstanding case for **VideoSAGE**, showing significantly higher performance on its respective correlation metrics.

Following the same reasoning as in section 4.2, we are comparing PGL-SUM [1] and **VideoSAGE** against ground truth scores. In the case of the predicted summaries, we are comparing against a (GT) summary, generated from the ground truth scores, as opposed to comparing against all user provided summaries. Correlation results for Video_16 in Table 3 show how PGL-SUM [1] is slightly better than **VideoSAGE**. Such results can be qualitatively validated on Figure 5 in which PGL-SUM [1] predicts the GT summary better. In this particular case, better correlation metrics also produced better F1-Score, however, that's not always the case.

Video_1 on Table 3 shows how PGL-SUM [1] have higher F1-Score than **VideoSAGE** even though their correlation results are notably lower than those obtained by **VideoSAGE**. We see from Figure 4 how PGL-SUM [1] clearly fails to predict the importance scores on Video_1 and how the predicted summary does not resemble that of the ground truth scores. Yet, its F1-Score gets benefited from

averaging over multiple user summaries.

Figure 3. TVSum/**Video_16**: Snapshots comparing shots from **VideoSAGE** (ours) predicted summary (Top) and a GT summary build from ground truth scores (Bottom).

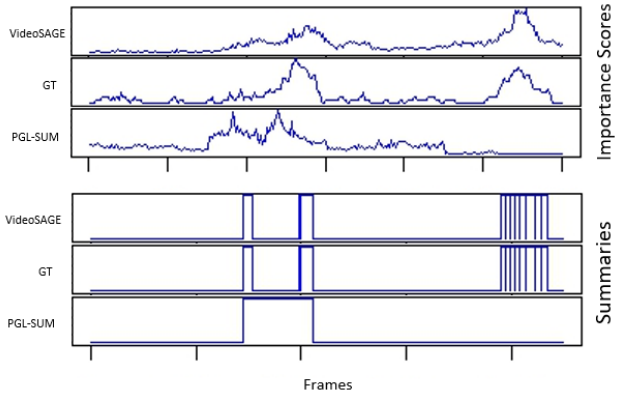

Figure 4. SumMe/**Video_1**: Comparison of importance scores and selected summary segments for **VideoSAGE** (ours), a ground truth, and PGL-SUM [1].

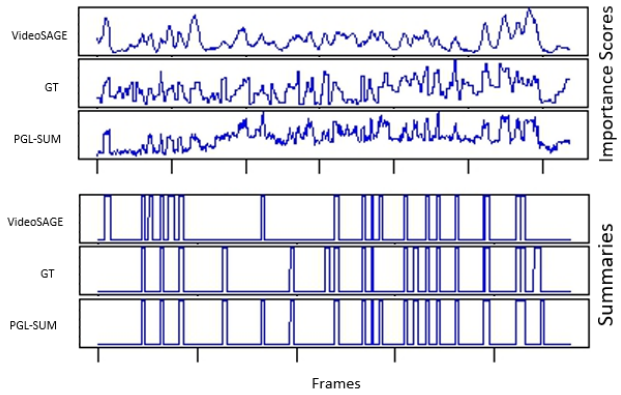

Figure 5. TVSum/**Video_16**: Comparison of importance scores and selected summary segments for **VideoSAGE** (ours), a ground truth, and PGL-SUM [1].

4.5. Parameters Setup

As explained in the Experiments section of Section 4.1 we utilizeRayTune's [11] grid search to find a proper set of hyper-parameters. Figure 6 shows the effect of different choices of the learning rate (lr) and different T distance values for the number of T forward and T backward connections per node in the graph representation on the Kendall's τ value for videos from TVSum [25] . These results are the average of 10 repeated experiments per split, totalling to 50 experiment runs per each combination of parameters.
The graph representation with best results was achieved for T value of 5. And the best learning rate is between lr=0.02 and lr=0.002. Bigger learning rates than that would result in a very unstable training. In fact, the standard deviation for T=5 and lr=0.02 was 0.061 while the standard deviation for T=5 and lr=0.002 was 0.051. Kendall's τ in those two configurations is virtually the same, so the learning rate with more stable results was chosen over the other.

Table 3. Comparing results for PGL-SUM [1] and **VideoSAGE** on two specific videos from SumMe and TVSum, showing its F1-Score and its Kendall's τ and Spearman's ρ correlation coefficients. * denotes reproduced results.

Model	SumMe/**video_1**			TVSum/**video_16**		
	F1	τ	ρ	F1	τ	ρ
PGL-SUM*	**69.69**	0.08	0.10	**68.54**	**0.32**	**0.47**
VideoSAGE (ours)	64.43	**0.47**	**0.60**	62.16	0.31	0.46

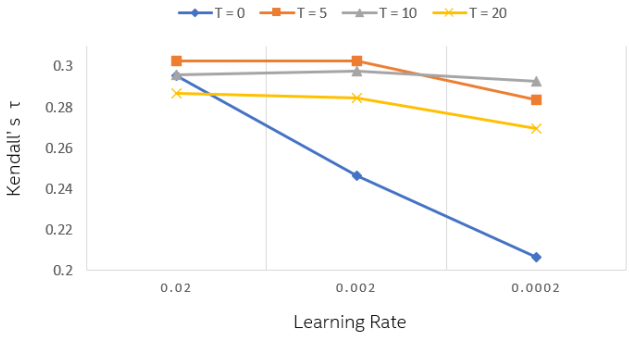

Figure 6. Kendall's τ correlation results on TVSum [25] for different choices of learning rate and number of graph edges (T) per node.

4.6. Profiling

All experiments including PGL-SUM [1], A2Summ [7] and **VideoSAGE** were run on an Intel(R) Core (TM) i9-12900k (3.2 GHz) with 32 GB of memory RAM. The evaluation experiments were run single threaded (for fair comparison) and profiled with PyTorch Profiler. Memory allocation details were extracted by using profiler's *export_memory_timeline* command. Results on table 4 are the averages of profiling the inference step during the respective normal evaluation runs of each models.
In Table 4, we can see how **VideoSAGE** can do inference an order of magnitude faster than the SOTA while requiring

Table 4. Comparing profiling results during inference on A2Summ [7], PGL-SUM [1] and **VideoSAGE**. * denotes reproduced results.

Model	Average Inference Time (ms)	Parameters' Memory (MB)	Max Memory Allocated (MB)
PGL-SUM*	113.79	36.02	55.17
A2Summ*	120.59	9.60	50.56
VideoSAGE (ours)	**23.55**	**3.52**	**19.27**

less than 2/5th of the memory allocated by PGL-SUM [1] or A2Summ [7].

5. Conclusions

We formulate the video summarization task as a binary node classification problem on graphs constructed from input videos. We first convert an input video to a graph where each node corresponds to a video frame and nodes within a specified temporal distance are connected to each other. We show that this structured sparsity leads to comparable or improved results on video summarization datasets while capable of performing one order of magnitude faster inference with only a fraction of the memory usage comparing with existing methods.

Acknowledgement We thank Kyle Min, who is the developer of **GraVi-T**, for his valuable feedback and comments.

References

[1] Evlampios Apostolidis, Georgios Balaouras, Vasileios Mezaris, and Ioannis Patras. Combining global and local attention with positional encoding for video summarization. In *2021 IEEE International Symposium on Multimedia (ISM)*, pages 226–234, 2021. 2, 3, 4, 5, 6, 7

[2] Evlampios Apostolidis, Georgios Balaouras, Vasileios Mezaris, and Ioannis Patras. Summarizing videos using concentrated attention and considering the uniqueness and diversity of the video frames. In *Proceedings of the 2022 International Conference on Multimedia Retrieval*, page 407–415, New York, NY, USA, 2022. Association for Computing Machinery. 2

[3] Jiri Fajtl, Hajar Sadeghi Sokeh, Vasileios Argyriou, Dorothy Monekosso, and Paolo Remagnino. Summarizing videos with attention. In *Computer Vision – ACCV 2018 Workshops*, pages 39–54, Cham, 2019. Springer International Publishing. 2

[4] Shijie Geng, Peng Gao, Chiori Hori, Jonathan Le Roux, and Anoop Cherian. Spatio-temporal scene graphs for video dialog. *arXiv e-prints*, pages arXiv–2007, 2020. 2

[5] Junaid Ahmed Ghauri, Sherzod Hakimov, and Ralph Ewerth. Supervised video summarization via multiple feature sets with parallel attention, 2021. 2

[6] William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs, 2018. 3

[7] Bo He, Jun Wang, Jielin Qiu, Trung Bui, Abhinav Shrivastava, and Zhaowen Wang. Align and attend: Multimodal summarization with dual contrastive losses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 4, 5, 6, 7

[8] Zhong Ji, Kailin Xiong, Yanwei Pang, and Xuelong Li. Video summarization with attention-based encoder-decoder networks, 2018. 2

[9] Hao Jiang and Yadong Mu. Joint video summarization and moment localization by cross-task sample transfer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16367–16377, 2022. 2

[10] Yunjae Jung, Donghyeon Cho, Dahun Kim, Sanghyun Woo, and In-So Kweon. Discriminative feature learning for unsupervised video summarization. *ArXiv*, abs/1811.09791, 2018. 2

[11] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018. 4, 6

[12] T. Darrell. M. Narasimhan, A. Rohrbach. Clip-it! language-guided video summarization. In *NeurIPS*, 2021. 2

[13] Li Mi, Yangjun Ou, and Zhenzhong Chen. Visual relationship forecasting in videos. *arXiv preprint arXiv:2107.01181*, 2021. 2

[14] Hayko Riemenschneider & Luc Van Gool Michael Gygli, Helmut Grabner. Creating summaries from user videos. In *European Conference on Computer Vision(ECCV)*, 2014. 2, 3, 4, 5

[15] Kyle Min. Intel labs at ego4d challenge 2022: A better baseline for audio-visual diarization. *arXiv preprint arXiv:2210.07764*, 2022. 2

[16] Kyle Min. Sthg: Spatial-temporal heterogeneous graph learning for advanced audio-visual diarization. *arXiv preprint arXiv:2306.10608*, 2023. 2

[17] Kyle Min, Sourya Roy, Subarna Tripathi, Tanaya Guha, and Somdeb Majumdar. Intel labs at activitynet challenge 2022: Spell for long-term active speaker detection. *The ActivityNet Large-Scale Activity Recognition Challenge*, 2022. https://research.google.com/ava/2022/S2_SPELL_ActivityNet_Challenge_2022.pdf. 2

[18] Kyle Min, Sourya Roy, Subarna Tripathi, Tanaya Guha, and Somdeb Majumdar. Learning long-term spatial-temporal graphs for active speaker detection. In *European Conference on Computer Vision*, pages 371–387. Springer, 2022. 2, 3, 4

[19] Tushar Nagarajan, Yanghao Li, Christoph Feichtenhofer, and Kristen Grauman. Ego-topo: Environment affordances from egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 163–172, 2020. 2

[20] Mayu Otani, Yuta Nakashima, Esa Rahtu, and Janne Heikkilä. Rethinking the evaluation of video summaries. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7588–7596, 2019. 5

[21] Jungin Park, Jiyoung Lee, Ig-Jae Kim, and Kwanghoon Sohn. Sumgraph: Video summarization via recursive graph modeling. *ArXiv*, abs/2007.08809, 2020. 2

[22] Mandela Patrick, Yuki M Asano, Bernie Huang, Ishan Misra, Florian Metze, Joao Henriques, and Andrea Vedaldi. Space-time crop & attend: Improving cross-modal video representation learning. *arXiv preprint arXiv:2103.10211*, 2021. 2

[23] Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid. Category-specific video summarization. In *ECCV 2014 - European Conference on Computer Vision*, 2014. 4

[24] Amir Shirian, Subarna Tripathi, and Tanaya Guha. Learnable graph inception network for emotion recognition. *IEEE Transactions on Multimedia*, 2020. 2

[25] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. Tvsum: Summarizing web videos using titles. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5179–5187, 2015. 2, 4, 5, 6

[26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 3, 5

[27] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015. 4

[28] Reuben Tan, Huijuan Xu, Kate Saenko, and Bryan A Plummer. Logan: Latent graph co-attention network for weakly-supervised video moment retrieval. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2083–2092, 2021. 2

[29] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Trans. Graph.*, 38(5), 2019. 3

[30] Yubo Zhang, Pavel Tokmakov, Martial Hebert, and Cordelia Schmid. A structured model for action detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9975–9984, 2019. 2

[31] Bin Zhao, Xuelong Li, and Xiaoqiang Lu. Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7405–7414, 2018. 2

[32] Bin Zhao, Haopeng Li, Xiaoqiang Lu, and Xuelong Li. Reconstructive sequence-graph network for video summarization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1, 2021. 2

[33] Kaiyang Zhou, Yu Qiao, and Tao Xiang. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward, 2018. 2

[34] Wencheng Zhu, Jiwen Lu, Jiahao Li, and Jie Zhou. Dsnet: A flexible detect-to-summarize network for video summarization. *IEEE Transactions on Image Processing*, 30:948–962, 2021. 2

[35] Wencheng Zhu, Yucheng Han, Jiwen Lu, and Jie Zhou. Relational reasoning over spatial-temporal graphs for video summarization. *IEEE Transactions on Image Processing*, 31: 3017–3031, 2022. 2