# Progressive Video Summarization via Multimodal Self-supervised Learning

Haopeng Li[1], Qiuhong Ke[3], Mingming Gong[2], Tom Drummond[1]
[1]School of Computing and Information Systems, the University of Melbourne
[2]School of Mathematics and Statistics, the University of Melbourne
[3]Department of Data Science & AI, Monash University
haopeng.li@student.unimelb.edu.au, {mingming.gong, tom.drummond}@unimelb.edu.au,
qiuhong.ke@monash.edu

**Category:** Pets & Animals
**Search Query:** Pet
**Title:** 40 Biggest and Most Unique Pets People Actually Own
**Description:** Some people like to have pets that they can put in their bags. However, some like pets that can easily overpower them. Take the family that took in a giant hippo, for example! Or even the family with a Dane as tall as 3 feet! …

**Category:** Sport
**Search Query:** Winter Sport
**Title:** 20 FUNNY MOMENTS IN WINTER SPORTS
**Description:** 20 FUNNY MOMENTS IN WINTER SPORTS

Figure 1. Two examples from the YTVT dataset. Five sampled frames and the text information of the video are presented. The text information of the video includes *category*, *search query*, *title*, and *description*.

## 1. More Examples from YTVY

Figure 1 shows more video examples from the proposed YTVY. As we can see from the examples, the videos from YTVY involve various topics such as *Pets & Animals* and *Sport*. For each video, we provide the text information including *topic*, *search query*, *title*, and *description*. Note that the presented text information is directly obtained from YouTube without pre-processing.

## 2. Residual Connection in Progressive Video Summarization

In the proposed progressive video summarization, the input of the $n$-th stage is computed as the weighted enhancement of the input of the previous stage based on the output of the previous stage, i.e.,

$$\boldsymbol{F}^n = \boldsymbol{F}^{n-1} * \boldsymbol{s}^{n-1} + \boldsymbol{F}^{n-1}, \tag{1}$$

where we add the input of the previous stage to the weighted features as residual connection. Intuitively, the input of the next stage can also be defined as the weighted features along (without residual connection) as follows,

$$\boldsymbol{F}^n = \boldsymbol{F}^{n-1} * \boldsymbol{s}^{n-1}. \tag{2}$$

Table 1. Comparison between the models with or without residual connection (ResCc).

| #Stages | ResCc | SumMe | | TVSum | |
|---|---|---|---|---|---|
| | | $\tau$ | $\rho$ | $\tau$ | $\rho$ |
| 2 | ✗ | 0.104 | 0.138 | 0.135 | 0.178 |
| | ✓ | 0.140 | 0.189 | 0.159 | 0.209 |
| 3 | ✗ | 0.096 | 0.128 | 0.142 | 0.187 |
| | ✓ | **0.166** | **0.224** | 0.162 | 0.212 |
| 4 | ✗ | 0.105 | 0.140 | 0.150 | 0.197 |
| | ✓ | 0.145 | 0.198 | **0.163** | **0.214** |

We conduct experiments to show the performance of these to formations of input in this part. The comparisons are shown in Table 1. Since we focus on the impact on the progressive mechanism, the self-supervised pretraining and the text information are not applied. As we can see from the results, the residual connection is of great importance to the performance of our progressive video summarization. Specifically, the impact of residual connection on SumMe is greater than that on TVSum. We believe the reason is that residual connection stabilize the training process on SumMe which consists of less training samples.

## 3. Different Score Prediction Manners

In this experiment, we show the impact of different score prediction manners on the performance. Three manners are tested: 1) using only the scores output by the last stage as the final scores (*Last*); 2) using the average of the scores output by all stages (*Avg.*); 3) using their step-wise multiplication as the final scores (*Mul.*). The formulations of the

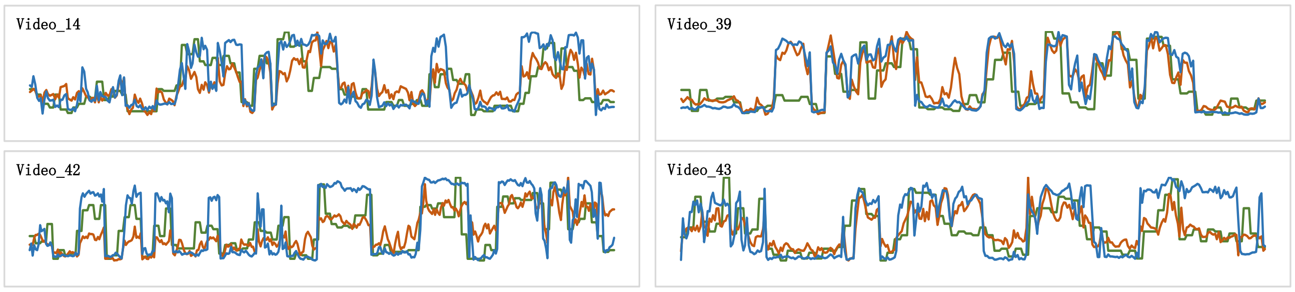

Figure 2. Visualization of the ground truth importance scores and the predicted ones by SSPVS and SSPVS+Text on TVSum. The green lines depict the ground truth scores, while the blue/red lines depict the predicted ones by SSPVS/SSPVS+Text.

Table 2. Comparison between different score prediction manners.

| #Stages | Manners | SumMe | | TVSum | |
|---|---|---|---|---|---|
| | | $\tau$ | $\rho$ | $\tau$ | $\rho$ |
| 2 | *Last* | 0.107 | 0.144 | 0.152 | 0.204 |
| | *Avg.* | 0.091 | 0.121 | 0.153 | 0.202 |
| | *Mul.* | 0.140 | 0.189 | 0.159 | 0.209 |
| 3 | *Last* | 0.069 | 0.091 | 0.161 | **0.214** |
| | *Avg.* | 0.077 | 0.103 | 0.162 | 0.212 |
| | *Mul.* | **0.166** | **0.224** | 0.162 | 0.212 |
| 4 | *Last* | 0.042 | 0.058 | 0.161 | 0.212 |
| | *Avg.* | 0.079 | 0.105 | 0.160 | 0.211 |
| | *Mul.* | 0.145 | 0.198 | **0.163** | **0.214** |

three manners are expressed respectively as follows:

$$\textit{Last}\text{: } \boldsymbol{s}^* = \boldsymbol{s}^N, \tag{3}$$

$$\textit{Avg.}\text{: } \boldsymbol{s}^* = \frac{1}{N}\sum_{n=1}^{N} \boldsymbol{s}^n, \tag{4}$$

$$\textit{Mul.}\text{: } \boldsymbol{s}^* = \boldsymbol{s}^1 \odot \boldsymbol{s}^2 \odot \cdots \odot \boldsymbol{s}^N, \tag{5}$$

The comparison results of the three score prediction manners are shown in Table 2. Similar to Section 2, the self-supervised pretraining and the text information are not applied. As shown in Table 2, on TVSum, different score prediction manners make little difference on the performance. However on SumMe, the score computed by multiplication outperforms other manners significantly. We assume the reason is that the multiplication makes the predicted score more discrimative for each frame. Considering the results on both datasets, we choose to use the multiplication of scores from all stages as the final score in our method.

## 4. Progressive Video Summarization with Text information

In this section, we show the impact of different manners of applying text information to progressive video summarization. We propose three manners: 1) add the text encoding in all stages (*All*); 2) add the text encoding only in the first stage (*First*); 3) add the text encoding only in the last stage (*Last*). The results of different manners are shown in Table 3. As we can see from the results, the three manners have different performance on SumMe. Specifically, when the text information is added in all stages or only the last stage, the results drop significantly. As for TVSum, the performance of three manners are similar, and adding the text information in the first stage gains slightly better results. Considering the results on two datasets, we assume the reason is that there exists a large domain gap between the encoded frames and encoded words in high stages, as we apply residual connection to the inputs before every stage. Hence, we only add the text information in the first stage.

## 5. Different Self-supervised Modeling

Our self-supervised framework consists of three loss functions to model coarse-grained multimodal correlations ($\mathcal{L}_1$), fine-grained multimodal correlations ($\mathcal{L}_2$), and temporal dependencies in videos ($\mathcal{L}_3$), respectively. Table 4 shows the results of the models pretrained by individual self-supervised loss functions. As the results show, each self-supervised model gains improvements for video summarization compared to the baseline model. Specifically, on SumMe, the temporal dependency modeling has the greatest impact, while on TVSum, the pretrained models obtain similar performance. By combining the three modeling for self-supervised learning, the pretrained model achieves the best results for video summarization on both datasets.

## 6. Impact of the Amount of Data for Self-supervised Pretraining

In this part, we show the impact of the amount of YTVT data for self-supervised pretraining on video summarization. Specifically, we randomly sample $\frac{1}{3}$ and $\frac{2}{3}$ of the videos from YTVT respectively for pretraining. We then fine-tune the two pretrained models on video summarization. The results are shown in Table 5. As we can see from the results, when we use $\frac{1}{3}$ of YTVT for pretraining, the

Table 3. Comparison of different manners of applying text information to video summarization.

| #Stages | Manners | SumMe | | TVSum | |
|---|---|---|---|---|---|
| | | $\tau$ | $\rho$ | $\tau$ | $\rho$ |
| 2 | *All* | 0.115 | 0.155 | 0.162 | 0.212 |
| | *First* | 0.174 | 0.235 | 0.163 | 0.214 |
| | *Last* | 0.088 | 0.116 | 0.161 | 0.210 |
| 3 | *All* | 0.126 | 0.171 | 0.171 | 0.227 |
| | *First* | **0.192** | **0.257** | 0.173 | 0.228 |
| | *Last* | 0.101 | 0.136 | 0.170 | 0.226 |
| 4 | *All* | 0.117 | 0.158 | 0.176 | 0.231 |
| | *First* | 0.175 | 0.237 | **0.181** | **0.238** |
| | *Last* | 0.096 | 0.130 | 0.174 | 0.229 |

Table 4. The results (Kendall's $\tau$ and Spearman's $\rho$) of different self-supervised loss functions.

| Loss | SumMe | | TVSum | |
|---|---|---|---|---|
| | $\tau$ | $\rho$ | $\tau$ | $\rho$ |
| Baseline | 0.137 | 0.187 | 0.141 | 0.185 |
| Coarse-grained ($\mathcal{L}_1$) | 0.147 | 0.196 | 0.149 | 0.196 |
| Fine-grained ($\mathcal{L}_2$) | 0.142 | 0.195 | 0.147 | 0.193 |
| Temporal ($\mathcal{L}_3$) | 0.149 | 0.201 | 0.147 | 0.194 |
| SSPVS | **0.154** | **0.207** | **0.151** | **0.199** |

Table 5. Impact of the percentage of YTVT used for pretraining.

| Proportion | SumMe | | TVSum | |
|---|---|---|---|---|
| | $\tau$ | $\rho$ | $\tau$ | $\rho$ |
| Baseline | 0.137 | 0.187 | 0.141 | 0.185 |
| 33.3% | 0.145 | 0.195 | 0.146 | 0.194 |
| 66.6% | 0.152 | 0.203 | 0.149 | 0.198 |
| 100% | **0.154** | **0.207** | **0.151** | **0.199** |

performance is improved considerably compared with the model without pretraining. As for the model pretrained with $\frac{2}{3}$ of YTVT, the improvements are even greater, but the results are close to those of whole YTVT. We believe that the capacity of the video encoder is the bottleneck and increase the model complexity could bring more improvement.

## 7. Impact of Different Types of Text Information

In this section, we show the impact of different types of information on video summarization. Since *search query* and *title* are available for all videos in TVSum, we conduct ablation studies on them for TVSum. The baseline model is pretrained on YTVT but no text information is used for video summarization. The results are shown in Table 6. As we can see from the results, *search query* outperforms *title* for video summarization on TVSum. We believe the reason is that *search query* provides more high-level semantic guidance which is of great importance to finding the most meaningful shots in videos.

Table 6. Impact of different types of text information.

| | Baseline | Query | Title | All |
|---|---|---|---|---|
| $\tau$ | 0.151 | 0.156 | 0.153 | 0.157 |
| $\rho$ | 0.199 | 0.205 | 0.202 | 0.206 |

Table 7. Comparisons of number of parameters and runtime of different video summarization models.

| Models | #Param. (M) | Runtime (s) |
|---|---|---|
| Bi-LSTM | 50.4 | 3.45 |
| Transformer | 75.6 | 0.83 |
| SSPVS(1) | | **0.66** |
| SSPVS(2) | **37.8** | 0.93 |
| SSPVS(3) | | 1.19 |
| SSPVS+Text(1) | | 1.86 |
| SSPVS+Text(2) | 148.7 | 2.16 |
| SSPVS+Text(3) | | 2.41 |

## 8. Computational Complexity Analysis

In this section, we conduct computational complexity analysis on various summarization models, including Bi-LSTM (with 2048D hidden state), the standard Transformer (with six layers), and our models with 1, 2, and 3 stages. The number of parameters and runtime of each model are shown in Table 7. Note that the runtime is measured by processing 100 videos of 200 frames, and the number of words in the text information is fixed to 50. As we can see from the results, our model has less parameters compared to Bi-LSTM and the standard Transformer. Since all stages share the same video encoder, the number of parameters is almost a constant with the increase in the number of stages. After exploiting the text information, the size of our model becomes much larger due to BERT. In terms of runtime, Bi-LSTM costs much more time compared to Transformer-based models. Besides, the time consumption of our model of less than three stages is close to the standard Transformer. Furthermore, with the text information, our model costs more time but is still efficient than Bi-LSTM. In summary, the computational cost of our methods is acceptable for real applications.

## 9. Visualization of Score Prediction

The predicted importance sores by SSPVS/SSPVS+Text and the ground truth ones are visualized in Figure 2. The four videos in the figure are from TVSum. As we can see from the figure, the predicted curves fit the ground truth ones well, which means the proposed methods can effectively model the temporal dependencies and the relative importance among frames. Specifically, SSPVS+Text (red) shows higher accuracy than SSPVS (blue) in the prediction of importance scores, which means the text information further benefits the process of video summarization.

## 10. Limitations

Our framework follows the typical pipeline of self-supervised learning: pretraining the video encoders and text encoders on YTVT without human annotations, and then fine-tune the encoders on the downstream video summarization with supervised learning. However, such a multi-stage training strategy is complicated compared to most video summarization methods. In the future work, we aims to fine-tine only part of the encoders on video summarization, or even directly apply the pretrained models to video summarization without fine-tuning (i.e., zero-shot video summarization), while maintaining the high performance, and thus the whole process is greatly simplified.