ADOPTING SELF-SUPERVISED LEARNING INTO UNSUPERVISED VIDEO SUMMARIZATION THROUGH RESTORATIVE SCORE.

Mehryar Abbasi, Student Member, IEEE, Parvaneh Saeedi, Member, IEEE

School of Engineering Science, Simon Fraser University, Burnaby, BC, Canada

ABSTRACT

In this paper, we present a new process for creating video summaries in an unsupervised manner. Our approach involves training a transformer encoder model to reconstruct missing frames in a video in a self-supervised way using the partially masked video as input. We then introduce an algorithm that utilizes the above-trained encoder to generate an importance score for each frame. Such frame importance scores are used to create the summary of the video. We show that the reconstruction loss of the model for a video with masked frames correlates with the representativeness of the remaining frames in the video. We validate the effectiveness of our approach on two benchmark datasets of TVSum and SumMe. We demonstrate that it outperforms state-of-the-art (SOTA) methods. Additionally, our approach is more stable during the training process compared to SOTA techniques based on generative adversarial learning. Our source code is publicly available[1].

Index Terms— Unsupervised Video Summarization, Unsupervised Learning, Self-supervised Learning, Self-attention Encoders.

1. INTRODUCTION

Video summarization is the task of creating a shortened version of a video that captures the most essential and informative content of the original video [1]. This can be done by extracting keyframes (also known as video skimming) or creating a series of video snippets (also known as video storyboarding) [1]. A general guideline for video summarization is that the summary should not be longer than 15% of the original video's length. This ensures that the summary can capture the most critical aspects of the video while still being concise and easy to watch. In recent years, there has been a shift towards using deep learning methods for automated video summarization [2]. Many of these approaches, however, rely on human-generated ground-truth labels to train their models [3–5], which can be time-consuming and subjective. In addition, such subjectivity implies that a video could have multiple possible summaries. As a result, there has been a focus on developing unsupervised video summarization methods without ground-truth labels for training [6–16].

A representative video summary enables a viewer to infer the original content of the video with less effort, time, and resources. Most unsupervised video summarization algorithms are built on such a principle [6, 9–16]. These methods use Generative Adversarial Networks (GAN) to create a representative summary that encapsulates the original content of the video. They, generally, train multiple deep models alongside the summarizer to imitate a viewer and decide whether the summary accurately represents the original video. The operational approach of these methods usually includes a summarizer, a generator, and a discriminator. The summarizer produces frame importance scores and creates a video snippet that includes only high-scoring frames. Using the video summary and the original input video, the generator creates two new videos, one from the original input video and one from the summary. The discriminator inspects the generator's outputs to identify which one of them was created from the summary. The training is performed in an adversarial manner. The summarizer is trained to create summaries that mislead the discriminator. The generator is trained to generate outputs that are similar to each other. Meanwhile, the discriminator is trained to make the distinction. The training process is not only highly complex but could be unstable.

[6] was the first published work to train a Long Short-Term Memory (LSTM)-based keyframe selector through an adversarial learning process. [14] improved the LSTM-based GAN model by improving the loss functions and optimization steps. [16] attempted to maximize the preservation of information in a video's summary by creating a more restrictive discriminator. [9] tackled the training difficulties for longer videos by adding a video decomposition stage. It broke each video into smaller, non-overlapping packs of consecutive frames and uniformly sampled strides before passing them to the summarizer network. The video frame scores were generated via the recomposition of the outputs for all packs and strides. Both [13, 15] tried to improve the algorithm presented in [6] by adding a frame score refinement stage on the summarizer's output. [15] employed an attention module that gradually edited each frame's score based on current and previous frames. [13] embedded an Actor-Critic model that adjusted the frames' scores in a non-sequential order based on past and future frames and previously made adjustments.

Training instability is an inherent problem among the GAN-based unsupervised summarizing networks [7]. Therefore, some approaches incorporated reinforcement learning with custom handcrafted reward functions [7, 17–19]. Such reward functions measure specific properties that must exist in an optimal video summary. [7] proposed a two-part reward function, named Diversity-Representativeness, that included measures of diversity and representativeness. The diversity measure quantified dissimilarities between frames of the summarized video. While the representativeness measure evaluated the visual representation (or the similarity) of the selected frames to the entire video. The goal was to train a model that creates summaries consisting of diverse representative frames of various parts of the video. [17] pointed out that most of the existing video summarization methods ignore inherent Spatio-temporal patterns in video data. They proposed a temporal segment network to create temporal and spatial scores for all video frames.

[1]https://github.com/mehryar72/RS-SUM

The average values of the temporal and spatial scores for the selected summary frames were used as reward values to train the summarizer network. [18] combined the diversity-representativeness reward presented in [7] with a jointly trained video reconstructor. They were able to effectively use both Award-based and Adversarial-based training processes. The reconstructor model was trained alongside the summarizer that included a diversity-representativeness reward in an adversarial manner.

Many of the previously mentioned video summarization methods use LSTM-based models. Unfortunately, they could suffer from issues including vanishing and exploding gradients [20]. In contrast, transformer models, which use self-attention mechanisms, have been successful in language processing tasks [21]. Some previous unsupervised video summarization methods have incorporated self-attention modules or transformer encoders into their LSTM-based models [10–12], while a few have used standalone transformer encoders [8, 19]. These methods primarily focused on swapping LSTM-based models for self-attention encoders. Despite utilizing self-attention encoders, these methods still relied on reward-based training using traditional rewards such as representative/diversity and length regularization cost [7, 14]. While these methods may have some success, it is clear that a more comprehensive and innovative approach is needed to truly advance the field.

Our proposed method in this paper is distinct from previous works in the field. We offer a self-supervised training approach that trains an encoder to reconstruct missing video sections using a rule-based masking operation. This process helps the model find and utilize long-term relations between frames instead of simply interpolating the missing information. Such training aims to create a model that can reconstruct an entire video from a given summary. We also introduce a pipeline that converts the trained model's reconstruction loss values into frame importance scores based on the idea that a good summary should enable good reconstruction of the original video. Our contributions therefore are:

1. A self-supervised encoder training method that produces a video reconstructor to regenerate a video from its summary.
2. A novel automated pipeline that uses a trained reconstructor with a new restorative score function to calculate frame-level importance scores and generate restorative video summaries.

2. APPROACH

The effectiveness of a video summary can be evaluated by comparing it to the original video using a video generator model. Specifically, the closer the reconstructed video produced by the model is to the original video, the more representative (or of higher quality) the summary is considered to be. Therefore, we use a high-performing video generator to assign an importance score to the frames of a summary based on how well they capture the original video's content. The first step is to use self-supervised training to create a video generator model that can reconstruct videos accurately. Next, we use the trained model and a reconstruction loss metric to score the individual frames of a video. We can then generate a video summary for evaluation based on these scores. The overall quality of the summary will depend on how well the most critical frames, as determined by the scoring algorithm, can capture the content of the original video. In this section, we will describe the multi-head self-attention encoder that we use in our approach, and the self-supervised training process and its parameters. We will also outline the algorithm that utilizes the trained encoder model to generate frame scores and how it is used to create a video summary.

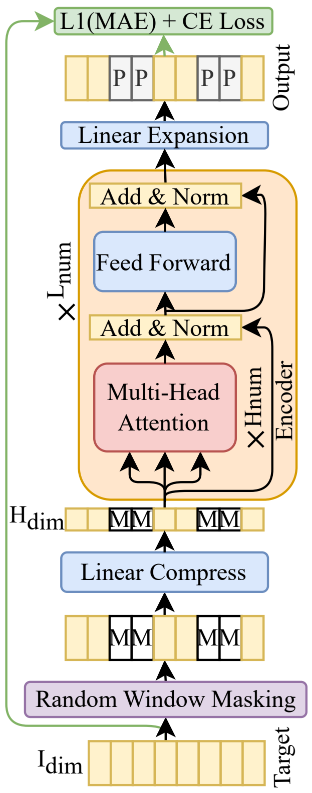

Fig. 1. Encoder model and the self-supervised training process.

2.1. Model Architecture

Fig. 1 displays the block diagram of our model. We use a single-layer multi-head self-attention encoder from a transformer model presented in [21]. This encoder is composed of two sublayers followed by a normalization layer. The first sublayer is a multi-head attention module. The second sublayer is a fully connected feed-forward network consisting of two linear transformations with a GeLU activation between them. For ease of implantation, we used the encoder architecture introduced in [22]. Our model also includes two Fully connected linear compress and expansion layers that convert I_{dim} to H_{dim} and back. I_{dim} presents the dimension of feature arrays of the input video frames (1024 for the two test benchmark datasets of TVSum and SumMe) and H_{dim} presents the dimension of the compressed (hidden) vectors, 512. H_{num} and L_{num} are the number of attention heads and encoder layers, respectively. H_{num} is set to 8, and L_{num} is set to 1.

2.2. Self-supervised video reconstruction training using random window maskeding

The video reconstructor model is trained in a self-supervised manner using a reconstruction loss that compares the original video input with its reconstructed version. This means that the model can be trained without any external labels, using only the original video as the ground truth. The process involves dividing the video into multiple samples and applying masks at different random positions. The following steps outline the specific procedure for this process.

The first step of the image sequence split could be performed in two ways: sequential split and dilated split. In a sequential split, ev-

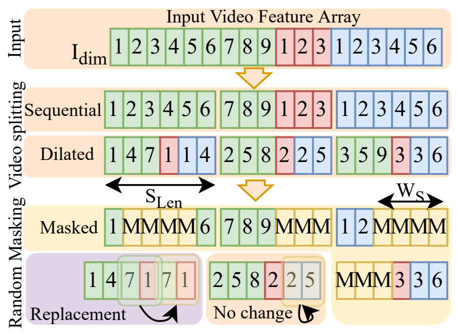

Fig. 2. Video splitting and random masking algorithm.

ery S_{Len} number of sequential frames is selected as one sample. The starting position for each split shifts by $\triangle$ during each epoch. $\triangle$ is a random number in the range of $\pm[0 \quad S_{Len}/2]$ and is independent of other splits. In the dilated split, each video is sampled into sections of S_{Len} frames with a dynamic dilation that changes based on the video's length. The splitting process is illustrated in Fig. 2.

It is crucial to explain the concept of shots to comprehend the second step of the random masking operation. A shot refers to a continuous sequence of similar frames in a video. A representation of an input segment is displayed at the top of Fig.2, where each shot is represented by a unique color. The shot boundaries are generated using Kernel Temporal Segmentation (KTS) [23] and used in the implementation of the second step of the random masking operation.

The second step of the process, random masking, selects a portion of each segment randomly for masking. Three rules govern the selection process. First, the number of chosen frames is set to $M_R\%$ of the size of each segment. Second, for each selected frame, a maximum of $W_s - 1$ additional frames from the same shot may be selected (forming a "window"). The selection may be limited either by the shot boundary or the window size (W_s). Third, not all of the chosen frames will be masked. Each selected window has an 80% probability of being masked, a 10% chance of being replaced with another unselected portion of the same size (replacement), and a 10% chance of remaining unchanged (No change). This is to enforce a dynamic range for the masking ratio. The specific above numbers are based on studies in self-supervised learning in NLP [24].

Finally, after a segment is prepared, it is passed to the model for reconstruction. The model then has to reconstruct any masked or replaced frames. The content of the reconstructed frames are then compared to their original content to generate a loss value. The loss function used in this process is a combination of the Mean Absolute Error (MAE) and the Cosine Embedding (CE) loss functions (s and $\hat{s}$ in Eq. 1 represent the original and reconstructed frame embeddings.).

$$\mathcal{L}_{\cos} = 1 - \frac{\mathbf{s} \cdot \hat{\mathbf{s}}}{\|\mathbf{s}\|_2 \cdot \|\hat{\mathbf{s}}\|_2}, \; \mathcal{L}_{\text{L1}} = \frac{1}{I_{dim}} \sum_{i=1}^{I_{dim}} |\mathbf{s_i} - \hat{\mathbf{s}}_i| \qquad (1)$$

2.3. Frame Score Generation

In this section, we explain our proposed iterative random masking approach that is used for frame importance score generation that uses the self-supervised trained model to rank the significance of each frame. Our scoring method quantifies the ability to reconstruct the original video from its summary using a restorative score function. The restorative score quantifies how good the frames are at reconstructing the original video from its summary if they are included in the summary frames. This pipeline repeats the following process $N = 30$ times. In each iteration:

1. The input video is divided into multiple segments of duration S_{Len}, using the same method described in Section 2.2 and shown in Fig. 2, with the exception that $\triangle$, the random split's starting position, is set to 0.
2. A random percentage ($\sigma\%$) of the frames in each segment is then selected and masked (Window size W_s is set to 1). The partially masked segment is passed to the trained model (σ has a similar function to the length regularization factor used in some SOTA methods such as [8, 13–16]).
3. The reconstructed segment is compared with the original (unmasked) video segment using"CE+MAE" loss function.
4. A restorative score is generated using the $Sigmoid(-Loss)$ operation and is assigned to the non-masked frames.

After all N iterations are complete, a score is calculated for each frame using the average of all scores assigned to it over N iterations. Algorithm 1 depicts the frame score generation pipeline. In this algorithm, C is the counter for the number of times a single frame is assigned a score. Naturally, frames included in multiple segments will have more than one score. Therefore, in the last step, The average score for each frame is calculated.

Algorithm 1: Frame Score Generation Pipeline

```
Input: V = $[F_1, F_2, ...F_L]$
 Counter: C← $[c_1, c_2, ...c_L] \leftarrow [0, 0, 0, ...]$;
Importance Scores: I← $[i_1, i_2, ...i_L] \leftarrow [0, 0, 0, ...]$;
V is split into multiple segments $[s_1, s_2, ...]$;
S ← $[s_1, s_2, ...]$;
for $s \in S$ do
    for $n \leftarrow 1$ to $N$ do
        M ← RandomMasking(s,$\sigma$=0.85,$W_s$=1);
        $\hat{s}$ ← Model(M);
        Loss ← CE(s,$\hat{s}$) + MAE(s,$\hat{s}$);
        Score ← Sigmoid(-Loss);
        for $F \in s$ do
            if $F$ is not masked then
                $c_F \leftarrow c_F + 1$;
                $i_F \leftarrow i_F + Score$;
            end
        end
    end
end
Output: I← $\frac{I}{C}$
```

2.4. Summary Generation

To have a fair comparison with the reported literature, we used the same algorithm as the SOTA [6, 8–16] to convert the generated frame importance scores to a video summary. Generally, most summary generation methods create summaries by selecting the most important (informative) shots from a video. A shot's importance is determined by averaging the frame-level scores of all the frames that constitute that shot (shot-level importance score). The objective is to select as many as possible the highest-scored shots while staying within a specified length limit for the summary. Naturally, shorter shots with higher importance scores are more likely to be selected in this process. The typical rule for video summary generation is that the summary should be within 5% to 15% of the original video's length. Therefore, we set the length limit for the summary to 15%.

This selection step is a 0/1 Knapsack problem that is optimally solved using dynamic programming [25]. The developed solution to this problem is the resultant video summary.

3. EXPERIMENTS AND RESULTS

To ensure a fair comparison between our method and the SOTA, we used the same evaluation procedure reported by the SOTA [6, 8–16]. The details of this procedure and our results are presented next.

3.1. Datasets and Evaluation methods

The performance of our proposed method is evaluated on two benchmark datasets of SumMe [26] and TVSum [27]. The TVSum dataset is composed of 50 videos with durations of 1 to 11 minutes. Each video is annotated by 20 users by creating importance scores at the frame- and shot-level. Users gave a score of 1 to 5 to each frame or shot. The SumMe dataset comprises 25 videos with durations of 1 to 6 minutes. Each video is annotated by 15 to 18 annotators in the form of key shots selection. Annotators produced a video summary (length range set to 5 to 15% of the video length) by selecting the most informative shots (video fragments).

The primary evaluation algorithm used by most SOTA video summarization methods is the F-Score similarity measure. The F-Score measures the similarity between the generated video summary to that of the user-annotated summary by looking at the overlap between the User summary (U) and the Automated summary (A). Eq. (2) demonstrates the formula for the calculation of F-Score. In this equation, P and R are Precision and Recall and Len refers to the length/size of an array.

$$F = 2 \times 100 \times \frac{P \times R}{P + R}, \; P = \frac{A \cap U}{Len\,(A)}, \; R = \frac{A \cap U}{Len\,(U)} \tag{2}$$

For each video in SumMe and TVSum datasets, the video summarization algorithm's output is compared against all user-generated annotations. This comparison results in multiple F-score values according to the number of annotations. To obtain a single F-score, a reduction operation is performed. For TVSum, the established benchmark guideline is to take the average of all F-scores as the final result, whereas for SumMe, the final F-score is determined by selecting the maximum F-score among all evaluations. After generating the F-scores for all videos of each dataset, an overall F-score is calculated for each video summarization method by taking the average the F-scores of all videos in that dataset. We used the predefined 5-fold data splits (80% training, 20% test) of each dataset. The experiment is repeated 5-times (once for each split), and the average results are reported.

3.2. Implementation Setup

Following the standard setting utilized by the SOTA unsupervised video summarization methods [6, 8–16], we used available video feature arrays extracted by [28]. [28] generated the video feature arrays by (1) downsampling the input videos to 2 frames per second (fps), and (2) generating the 1024 dimensional output of GoogleNet's [29] penultimate layer for the sampled frames.

The self-supervised training step is performed over 250 epochs. M_R and W_S are set to 25% and 8, respectively. The batch Size is set to 256. We used AdamW optimizer with a Cosine learning rate scheduler. Fig. 3 depicts the learning rate curve during the self-supervised training. During the frame score generation step, σ is set to 85%, which translates to a summary length of 15%.

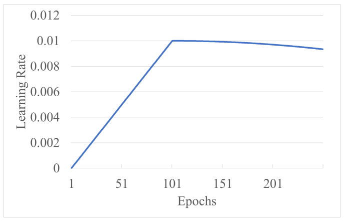

Fig. 3. Learning rate curve during the self-supervised training.

Table 1. F-Scores results. * highlights models with different σ for each dataset and ** methods with different σ for each fold.

Dataset	SumMe		TVSum		
Method	F-Score	Rank	F-Score	Rank	Avg Rank
SUM-GAN [6]	41.7	7	56.3	8	7.5
Cycle-Sum* [16]	41.9	6	57.6	7	6.5
DR-DSN* [7]	41.4	8	57.6	6	7
SUM-GAN-AAE* [14]	48.9	4	58.3	5	4.5
SUM-GAN-sl* [15]	47.8	5	58.4	4	4.5
CSNet [9]	51.3	2	58.8	3	2.5
AC-SUM-GAN* [13]	50.8	3	60.6	2	2.5
RS-SUM (ours)	**52.0**	1	**61.1**	1	1
Results with the different σ for each split					
CA-SUM** [8]	51.1	2	*61.4*	1	1.5
RS-SUM (ours)**	*52.5*	1	*61.4*	1	1

3.3. Performance Comparison

The performance of the proposed method, Restorative Score video summarizer (RS-SUM), is compared against the leading SOTA in Table 1. These results indicate that our model ranks first on both TVSum and SumMe datasets. Unlike most mentioned methods, RS-SUM uses the same masking ratio, σ%, for both datasets. Many SOTA methods, such as those indicated by $*$ in Table 1, use a term called the Length Regularization Factor [6–8, 13–16] to penalize the selection of a large number of key frames in the summary during training. These methods typically train multiple models with different σ values (between 0.05 to 0.95 with 0.05 steps) and only report the highest value for each dataset. This means that two different models are trained and tested on each dataset. In contrast, we use a unified value of 0.85 for both datasets.

Moreover, CA-SUM [8] trained multiple models with different σ values on each data fold and recorded the highest F-score for each data split with a different σ. This is unlike our method that keeps the parameters fixed for all presented results. As a result, we separated their results from the other methods in Table 1.

4. CONCLUSION

We proposed a new unsupervised video summarization pipeline that uses a stable self-supervised training method with a self-attention encoder. Our self-supervised training approach is designed to generate a model that can reconstruct a video from its summary. Our proposed iterative pipeline uses the trained encoder to assign importance scores to each frame. We demonstrate the effectiveness of our algorithm by testing it on two benchmark datasets of, TVSum and SumMe, using the same configurations as previous SOTA works. Our system achieves a new F-score record on both datasets. Our future work will focus on reducing the overhead cost during the inference stage caused by the iterative manner of frame score generation.

References

[1] S. Jadon and M. Jasim, "Unsupervised video summarization framework using keyframe extraction and video skimming," in *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)*, IEEE, 2020, pp. 140–145.

[2] E. Apostolidis *et al.*, "Video summarization using deep neural networks: A survey," *Proceedings of the IEEE*, vol. 109 (11), pp. 1838–1863, 2021.

[3] J. A. Ghauri, S. Hakimov, and R. Ewerth, "Supervised video summarization via multiple feature sets with parallel attention," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2021, 1–6s.

[4] W. Zhu *et al.*, "Relational reasoning over spatial-temporal graphs for video summarization," *IEEE Transactions on Image Processing*, vol. 31, pp. 3017–3031, 2022.

[5] E. Apostolidis *et al.*, "Combining global and local attention with positional encoding for video summarization," in *2021 IEEE International Symposium on Multimedia (ISM)*, IEEE, 2021, pp. 226–234.

[6] B. Mahasseni, M. Lam, and S. Todorovic, "Unsupervised video summarization with adversarial lstm networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 202–211.

[7] K. Zhou, Y. Qiao, and T. Xiang, "Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, 2018.

[8] E. Apostolidis *et al.*, "Summarizing videos using concentrated attention and considering the uniqueness and diversity of the video frames," ser. ICMR '22, Newark, NJ, USA: Association for Computing Machinery, 2022, pp. 407–415, ISBN: 9781450392389.

[9] Y. Jung *et al.*, "Discriminative feature learning for unsupervised video summarization," in *Proceedings of the AAAI Conference on artificial intelligence*, vol. 33, 2019, pp. 8537–8544.

[10] Y. Jung *et al.*, "Global-and-local relative position embedding for unsupervised video summarization," in *European Conference on Computer Vision*, Springer, 2020, pp. 167–183.

[11] Y.-T. Liu *et al.*, "Learning hierarchical self-attention for video summarization," in *2019 IEEE international conference on image processing (ICIP)*, IEEE, 2019, pp. 3377–3381.

[12] X. He *et al.*, "Unsupervised video summarization with attentive conditional generative adversarial networks," in *Proceedings of the 27th ACM International Conference on multimedia*, 2019, pp. 2296–2304.

[13] E. Apostolidis *et al.*, "Ac-sum-gan: Connecting actor-critic and generative adversarial networks for unsupervised video summarization," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 31 (8), pp. 3278–3292, 2020.

[14] E. Apostolidis *et al.*, "A stepwise, label-based approach for improving the adversarial training in unsupervised video summarization," in *Proceedings of the 1st International Workshop on AI for Smart TV Content Production, Access and Delivery*, 2019, pp. 17–25.

[15] E. Apostolidis *et al.*, "Unsupervised video summarization via attention-driven adversarial learning," in *International Conference on multimedia modeling*, Springer, 2020, pp. 492–504.

[16] L. Yuan *et al.*, "Cycle-sum: Cycle-consistent adversarial lstm networks for unsupervised video summarization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 9143–9150.

[17] N. Gonuguntla, B. Mandal, N. Puhan, *et al.*, "Enhanced deep video summarization network," BMVC, 2019.

[18] B. Zhao, X. Li, and X. Lu, "Property-constrained dual learning for video summarization," *IEEE transactions on neural networks and learning systems*, vol. 31 (10), pp. 3989–4000, 2019.

[19] U.-N. Yoon, M.-D. Hong, and G.-S. Jo, "Interp-sum: Unsupervised video summarization with piecewise linear interpolation," *Sensors*, vol. 21 (13), p. 4562, 2021.

[20] S. Li *et al.*, "Independently recurrent neural network (indrnn): Building a longer and deeper rnn," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5457–5466.

[21] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[22] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[23] D. Potapov *et al.*, "Category-specific video summarization," in *European conference on computer vision*, Springer, 2014, pp. 540–555.

[24] J. Devlin *et al.*, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[25] Y. Song *et al.*, "Tvsum: Summarizing web videos using titles," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5179–5187.

[26] M. Gygli *et al.*, "Creating summaries from user videos," in *European conference on computer vision*, Springer, 2014, pp. 505–520.

[27] Y. Song *et al.*, "Tvsum: Summarizing web videos using titles," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5179–5187.

[28] K. Zhang *et al.*, "Video summarization with long short-term memory," in *European conference on computer vision*, Springer, 2016, pp. 766–782.

[29] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.