

Unsupervised Video Summarization

Hanqing Li¹, Diego Klabjan², Jean Utke³

¹Department of Engineering Science and Applied Mathematics,
Northwestern University, Evanston, Illinois, USA

²Department of Industrial Engineering and Management Sciences,
Northwestern University, Evanston, Illinois, USA

³Allstate Insurance Company

Abstract

This paper introduces a new, unsupervised method for automatic video summarization using ideas from generative adversarial networks but eliminating the discriminator, having a simple loss function, and separating training of different parts of the model. An iterative training strategy is also applied by alternately training the reconstructor and the frame selector for multiple iterations. Furthermore, a trainable mask vector is added to the model in summary generation during training and evaluation. The method also includes an unsupervised model selection algorithm. Results from experiments on two public datasets (SumMe and TVSum) and four datasets we created (Soccer, LoL, MLB, and ShortMLB) demonstrate the effectiveness of each component on the model performance, particularly the iterative training strategy. Evaluations and comparisons with the state-of-the-art methods highlight the advantages of the proposed method in performance, stability, and training efficiency.

1 Introduction

On the internet, there is a seemingly endless stream of social media and sharing platforms carrying a sea of video content, which creates the need to navigate and locate valuable clips efficiently. One solution to this need lies in video summarization. Video summarization aids in browsing large and continually growing collections by synthesizing an overwhelming amount of information into an easily digestible form. In this paper, we propose an unsupervised learning model that automatically summarizes video. We name the model SUM-SR according to its summarization function and architecture containing a selector and a reconstructor.

Most research approaches the video summarization task in a supervised manner, using ground-truth annotations to guide the learning process, Apostolidis et al.[3]. Nonetheless, there are also several unsupervised approaches that are trained without the need of ground-truth data, eliminating the need for laborious and time-consuming annotation tasks. The competitive performance of some unsupervised methods and the limited availability of

ground-truth data suggest that unsupervised video summarization approaches have significant potential.

SUM-SR builds on a GAN-based method (SUM-GAN-AAE) by Apostolidis et al.[2] while removing the discriminator. Similar to SUM-GAN-AAE, the model has a selector for choosing key fragments from the original video and a reconstructor with an attention mechanism for reconstructing the original video from the video summary. However, instead of comparing the original video and the generated summary by an additional discriminator, which increases the complexity of training, we directly calculate the mean square error (MSE) between the embeddings of the two videos as the loss function. Compared to numerous loss functions in SUM-GAN-AAE, SUM-SR uses only the reconstruction and a regularization loss to guide the training process. We introduce an extra training step that separates the training of the reconstructor and the selector as they have different functionalities. Moreover, we extend this term-by-term training strategy to an iterative one that trains the two parts of the model alternately with multiple iterations. Such an approach further improves the model’s performance. Lastly, we design an unsupervised algorithm to select the best model after training. We test the performance of the model on two benchmark datasets: TVSum, Song et al.[13], and SumMe, Gygli et al.[7], as well as four datasets we created. The proposed model demonstrates better performance than previous state-of-the-art methods by 8.5% on average based on the per dataset best benchmark and 9.2% based on a single best benchmark.

Our contributions are as follows.

- We create a new framework for the task of unsupervised video summarization by comparing a generated summary with its original video only through a reconstructor network without using a discriminator.
- We introduce an extra training step for the reconstructor and an iterative training strategy to increase the performance of the model.
- We also design a function to select the best model after training in an unsupervised manner.

The rest of the paper is organized as follows. In the next section, we review previous research on unsupervised video summarization. In Section 3, we detail the proposed unsupervised deep learning approach. In Section 4, we present the experimental results and compare them to state-of-the-art methods. Finally, in Section 5, we conclude the paper.

2 Related Work

In recent years, there have been several approaches to automatic video summarization. These methods can be categorized into supervised and unsupervised based on the presence of ground-truth labels. In this section, we focus on presenting relevant literature on unsupervised approaches. If readers are interested in supervised video summarization, they are referred to the review by Apostolidis et al.[3].

For unsupervised learning, to overcome the challenge of no guidance from ground-truth labels for training, essential characteristics of a good video summary are tackled. Recent work in unsupervised video summarization tries to construct video summaries that are highly

representative of the original video content. The goal is to equate the original video with a good video summary. Based on this assumption, most approaches involve Generator-Discriminator architectures and the adversarial training mechanisms that guide the summarization component to produce a summary capable of a good reconstruction of the original video. Mahasseni et al.[12] first try to incorporate adversarial learning by adding a Variational Auto-Encoder and a discriminator on top of an LSTM-based keyframe selector. The model is trained by minimizing the distance between the latent representation of the original video and the summary generated by the selector. Following this work, Apostolidis et al.[5] propose a stepwise, label-based training approach for the adversarial module of the network by Mahasseni et al.[12] that improves the summarization performance. Apostolidis et al. also improve the model performance by editing the model structure in their two following works[2, 1]. In SUM-GAN-AAE[2], Apostolidis et al. replace the Variational Auto-Encoder with a deterministic attention-based Auto-Encoder. In AC-SUM-GAN[1], Apostolidis et al. embed an Actor-Critic model into the previous model to combine adversarial learning and reinforcement learning and reformulate the selection of important video fragments as a sequence generation task. In their most recent work, Apostolidis et al.[4] replace the GAN structure and adversarial learning with an attention mechanism that utilizes the uniqueness and diversity of each frame to create a summary. Moreover, Jung et al.[8], based on the model by Mahasseni et al.[12], develop a new two-stream network named Chunk and Stride Network (CSNet) that utilizes both local (chunk) and global (stride) temporal information of a video; meanwhile, they introduce a variance loss to increase the discrepancy of importance scores of frames and scene transitions to include dynamic information in videos. In their following work, Jung et al.[9] adopt a self-attention mechanism and relative position embedding for video input to handle long-term dependency in a video sequence. Kanafani et al.[10] further extend the CSNet, Jung et al.[8], to a model that takes multiple visual representations (GoogleNet and ResNet-50) extracted from a video as input to enhance the model performance.

Some other unsupervised approaches for video summarization use hand-crafted reward functions to quantify the presence of desired characteristics (such as representativeness and diversity) in the generated summary and train a network architecture for video summarization using reinforcement learning. Zhou et al.[18] use a simple LSTM-based network architecture and a pair of diversity and representativeness rewards to formulate video summarization as a sequential decision-making process. The diversity reward measures the dissimilarity among selected keyframes, while the representativeness reward measures the visual resemblance of the selected keyframes to the rest of the video frames. Gonuguntla et al.[6] use temporal segment networks introduced by Wang et al.[15] to extract spatial and temporal information from the video frames and train a video summarization architecture using a reward function that evaluates the preservation of the main spatiotemporal patterns in the produced summary. Zhao et al.[17] present a system that performs both video summarization and reconstruction, with the latter being used to assess how well the summary allows the viewer to infer the original video. The video summarization is learned based on the output of the video reconstruction process and trained models that evaluate the representativeness and diversity of the generated summary. Yaliniz et al.[16] use independent recurrent neural networks developed by Li et al.[11] to model the temporal dependence of video frames and learned summarization using rewards for representativeness, diversity,

and uniformity (temporal coherence) of the video summary. Phaphuangwittayakul et al.[6] propose a variation of the network architecture from Zhou et al.[18] that estimates the importance of frames by combining the representations of the video frames as the output of a bi-directional RNN and a self-attention mechanism.

Compared to GAN-based methods mentioned above[12, 5, 2, 1, 8, 9, 10], the proposed approach eliminates the discriminator, thereby simplifying the training steps and removing the potential risk of unstable training when using GANs, Zhou et al.[18]. We do not update every trainable weight in the model at each epoch, but we separate the training of the selector and the reconstructor to enhance their performance. The reinforcement learning methods[18, 6, 15, 17, 16, 11, 6] above employ hand-crafted reward functions which are hard to tailor and often lead to poor performance, Apostolidis et al.[1]. We let the model learn how to construct a good summary from the original video through a deep learning approach instead of optimizing hand-crafted reward functions through a reinforcement learning approach. The most recent research by Apostolidis et al.[4] contains neither GANs nor reinforcement learning. Nevertheless, they train their model only with the average distance between frame-level importance scores and a regulation hyperparameter, which is not directly related to creating a good summary and causes unstable model performance. In contrast, we include the reconstruction loss in the training process, which is built upon the assumption that a good summary could help recover the video. All previous works[12, 5, 2, 1, 8, 9, 10, 18, 6, 15, 17, 16, 11, 6, 1] select the best model based on its performance on the validation set, which needs true labels. We introduce a new method associated with the proposed model to select the best model in unsupervised fashion using only the reconstruction and sparsity losses on the validation dataset.

3 Model

This section explains the design and structure of the SUM-SR model and the training process. We describe in detail the model selection method and the function that generates a video summary from the output (importance scores) of the selector.

3.1 Model Structure

We subsample each video and use a pre-trained CNN to encode each frame. Each video is represented by a sequence of vectors $V = (x_1, x_2, \dots, x_n)$, where $x_i \in \mathbb{R}^d$ is the embedding of the frame i of video V . We treat the summarization task as a binary classification problem. For each frame, the model decides whether or not to include it in the summary and we view the probability of inclusion as the importance score for this frame.

The proposed model is composed of a selector and a reconstructor (see Figure 1). In the following sections, we denote the selector as *sNet* and the reconstructor as *rNet*. The selector has a linear layer to compress the input dimension from d to d_h , a bidirectional LSTM, and an output layer that maps the output h_i of the LSTM to an importance score p_i . The output layer first maps h_i to a two-dimensional vector \hat{h}_i and computes p_i by softmax

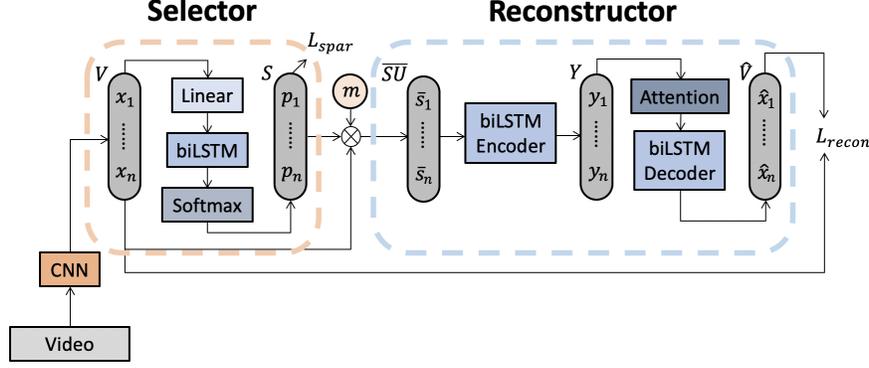


Figure 1: The proposed SUM-SR architecture.

function with temperature τ as follows

$$\begin{aligned}
 h_i &= \text{biLSTM}(\text{Lin}(x_i), h_{i-1}) \\
 \hat{h}_i &= \text{Lin}(h_i) \\
 p_i &= \text{softmax}(\hat{h}_i, \tau)_1.
 \end{aligned}$$

Then we define $S = (p_1, p_2, p_3, \dots, p_n)$ as the importance scores of video V . Given video V and importance scores S , we use a non-parametric function f to create a summary by selecting important frames. The output is a vector $A = (a_1, a_2, \dots, a_n)$ with binary entries $a_i \in \{0, 1\}$ that indicate whether the i -th frame is selected or not. From A , we build a summary $SU = (s_1, s_2, \dots, s_n)$, where $s_i = x_i$ if $a_i = 1$ and $s_i = m$ if $a_i = 0$. Vector m is a mask vector with dimension d . We explain more about f and m in later sections.

Since the function f is not differentiable, during training, we use a different method to create a trainable summary $\overline{SU} = (\bar{s}_1, \bar{s}_2, \dots, \bar{s}_n)$. Each entry \bar{s}_i is a weighted sum of x_i and m given by $\bar{s}_i = p_i \cdot x_i + (1 - p_i) \cdot m$.

The reconstructor of the model is an autoencoder with an attention block. Both the encoder and decoder are bi-directional LSTMs with the input to the encoder being \overline{SU} . Focusing on the attention block (denoted as m_attn), for any time step $i \in [2 : n]$, the attention block has access to the encoder output $Y = (y_1, y_2, \dots, y_n)$, and the previous hidden state of the decoder, z_{i-1} . We compute the attention energy vector e_i from Y and z_{i-1} by $e_i = Y^T W_b z_{i-1}$ where W_b is a trainable matrix. At time step $i = 1$, we use the last hidden state of the encoder h_e to calculate e_1 . Afterward, we apply a softmax function on e_i to get a normalized attention weight vector $w_i = \text{softmax}(e_i)$ and multiply w_i with the encoder's output to produce a context vector $y'_i = Y w_i$. The context vector y'_i and the previous output of the decoder are concatenated together to form the input to the decoder at time step i . Given \overline{SU} as the input, the reconstructor outputs the reconstructed video $\hat{V} = (\hat{x}_1, \dots, \hat{x}_n)$ as follows

$$\begin{aligned}
 Y &= \text{Encoder}(\overline{SU}) \\
 \hat{V} &= \text{Decoder}_{\text{atten}}(Y).
 \end{aligned}$$

$Decoder_{atten}$ is a bi-directional LSTM with attention block m_{attn} .

During training, we use two loss functions: 1) reconstruction loss, L_{recon} , and 2) regularization loss, L_{spar} . Following SUM-GAN-AAE, Apostolidis et al.[2], our goal is to train the selector to generate a summary that could be reconstructed to the original video through the reconstructor. We define the reconstruction loss as the Euclidean distance between the original video frame embeddings V and the reconstructed video frame embeddings \hat{V} based on $L_{recon} = ||V - \hat{V}||^2$. To avoid the trivial solution of selecting all frames, we introduce the Summary-Length regularization, Mahasseni et al.[12], which penalizes the model when it assigns high importance scores to a large number of frames. The regularization loss is computed by $L_{spar} = ||\frac{1}{n} \sum_{i=1}^n p_i - \sigma||$, where σ is a hyperparameter between 0 and 1. We train the model based on the loss function $L_{model} = L_{recon} + L_{spar}$.

3.2 Training Strategy

For inference, we keep only the selector to generate a video summary after training is complete. To make the training process more focused on updating the selector, we separate the training of the selector and the reconstructor. One iteration consists of training first only the reconstructor and then only the selector. We iterate several times. To prevent in the first iteration to train the selector with random reconstruction weights, we train the reconstructor first, see Figure 2. Our goal is to create a shorter video summary with length αL from a video with length L , where α is the summary rate in $(0, 1)$, and the reconstructor aims to reconstruct the original video from the shorter video. Thus, when training the reconstructor, we create such a shorter video by randomly replacing some of the vectors x_i from a video $V = (x_1, x_2, \dots, x_n)$ with a mask vector m to shorten the video length by masking information. Because our summary rate is α , we want to keep α fraction of frames and replace the rest with the mask vector m to get $V' = (x'_1, x'_2, \dots, x'_n)$, where $p(x'_i = x_i) = \alpha, p(x'_i = m) = 1 - \alpha$. We feed V' into the reconstructor to get a reconstructed video $\hat{V}' = (\hat{x}'_1, \hat{x}'_2, \dots, \hat{x}'_n)$, and train the reconstructor by $L_{recon} = ||V - \hat{V}'||^2$. Then, we train only the selector based on L_{model} in Section 3.1.

The mask vector m is also trainable. To train the mask vector, we develop two strategies. The first strategy is updating the mask vector m together with the reconstructor when training only the reconstructor in the first iteration. Another strategy is to train the mask vector m alone first. We first create a new reconstructor R (we use R only to train the mask vector) and initialize m to zero vector. Then, we randomly replace some vectors in V with m with probability $1 - \alpha$ to get V' and feed it into the reconstructor R . We train both R and m with the loss function $L_{mask} = \frac{1}{|\mathcal{D}|} \sum_{j \in \mathcal{D}} ||x'_j - \hat{x}'_j||^2$, where \mathcal{D} is the set of indices j such that $x'_j = m$ and \hat{x}'_j is an output from R . After this training process, the mask vector remains fixed during the reconstructor’s training step and the selector’s training step (the rest of R is discarded).

We define one reconstruction and selection as an iteration. To further improve the model performance, we train the model with multiple iterations. In each iteration, we initialize the model as the selected model from the previous iteration.

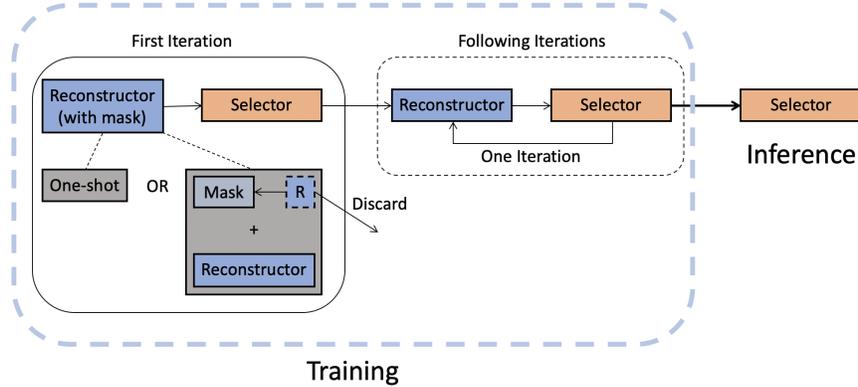


Figure 2: The training steps of SUM-SR. During training, an iteration includes one reconstruction and one selection. We train the mask vector only in the first iteration (if there are multiple iterations).

3.3 Summarization

After obtaining importance scores S , we use a non-parametric function $f(V, S)$ to generate a summary of video V . We first obtain video shots (a continuous clip of a video that contains multiple frames) with the KTS algorithm introduced in [2]. Then, we calculate the shot-level importance scores by averaging the frame-level importance scores of each shot. Finally, we generate the summary by maximizing the sum of the shot-level importance scores. Meanwhile, we ensure the summary length is shorter than α fraction of the original video length. We formulate this as the knapsack problem

$$\max_{\hat{A}} \sum_{i=1}^N \hat{a}_i \cdot \hat{p}_i, \text{ s.t. } \sum_{i=1}^N \hat{a}_i \cdot l_i \leq \alpha \cdot L, \hat{a}_i \in \{0, 1\},$$

where N is the number of shots, L is the length of the original video V , and \hat{p}_i is the shot-level importance score of the i -th shot. Binary variable \hat{a}_i indicates whether to include the i -th shot in the summary, and l_i is the length of the i -th shot. We define the shot-level summary vector as $\hat{A} = (\hat{a}_1, \hat{a}_2, \dots, \hat{a}_N)$.

3.4 Model Selection

Since the model is unsupervised, we need an unsupervised method to select the model for inference. In a single iteration, for all models from all training epochs, we generate the summary from the selector ($sNet_i$) according to Section 3.1 and obtain the reconstructed video using the reconstructor ($rNet_i$). Here $i = 1, \dots, E$ with E being the total number of epochs. Consider epoch i and video V_j in the validation dataset. We first calculate $S_{ij} = sNet_i(V_j)$, $A_{ij} = f(V_j, S_{ij})$. By using A_{ij} , we next generate SU_{ij} according to Section 3.1. Note that this is different from training, where \overline{SU} is used because of to the need for differentiability. Finally, we get the reconstruction video embedding by $\hat{V}_{ij} = rNet_0(SU_{ij})$. We use $rNet_0$ from the reconstructor-only training stage as it is expected to be of high quality. At this point, we have $L_{recon_{ij}} = \|\hat{V}_{ij} - V_j\|^2$ and $L_{spar_{ij}}$.

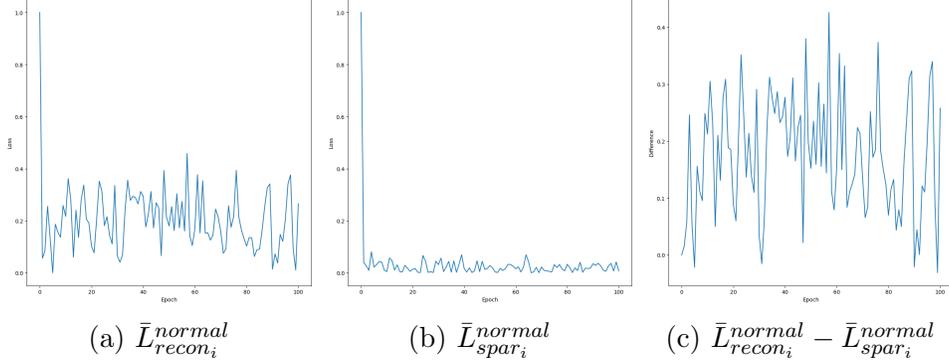


Figure 3: Normalized Loss Curve of one Experiment on TVSum

We follow by first averaging all samples in validation to get \bar{L}_{recon_i} and \bar{L}_{spar_i} , which are then separately scaled so that across all i the losses are in $[0, 1]$ to get $\bar{L}_{recon_i}^{normal}$ and $\bar{L}_{spar_i}^{normal}$. For epoch i , the selected model is

$$\operatorname{argmax}_i(\bar{L}_{recon_i}^{normal} - \bar{L}_{spar_i}^{normal}).$$

This selection method is decided based on experiment results and inspired by loss curves in Figure 3. After normalizing both \bar{L}_{recon_i} and \bar{L}_{spar_i} to $[0, 1]$, it is noteworthy that their difference is small because their initial values are large. Subsequently, as both losses converge towards a small value and oscillate in proximity to this converged state, the normalized difference starts increasing with oscillation. Based on the selection method, a model is selected after convergence, avoiding the trivial solution of a model with an extremely high reconstruction loss coupled with a minimal sparsity loss. The experiment result of the model performance in Section 4 also proves the solidity of the model selection method.

For the experiment without separating training of the reconstructor and selector, we first pick a reconstructor with the smallest validation reconstruction loss following the same expressions except that $rNet_i$ replaces $rNet_0$. This yields the best reconstruction model β . Finally, we repeat the previous model selection steps by using $rNet_\beta$ instead of $rNet_0$.

For the experiment with multiple iterations, we first pick a target model in each iteration using the aforementioned model selection method and pick the target model with the smallest reconstruction loss on the validation set.

4 Experiments

We select CA-SUM[4], AC-SUM-GAN[1], CSNet[8] and SUM-GAN-AAE[2] as benchmarks for performance comparison. According to previous works[4, 8, 1], CA-SUM performs the best on TVSum, CSNet performs the best on SumMe and AC-SUM-GAN is the best approach using reinforcement learning. We also include SUM-GAN-AAE[2] because our approach builds on it. All benchmarks use ground truth summary in model selection, but our approach uses the unsupervised model selection method. For a fair comparison, we use an unsupervised model selection method for each benchmark model. For CA-SUM[4], we use its proposed unsupervised model selection method by choosing the model with the smallest

model loss L_{reg} , as defined in [4], on the validation set. For AC-SUM-GAN[1], Apostolidis et al. mention a model selection method selecting the best model with the highest reward and simultaneously the smallest actor’s loss on the validation set. We follow this model selection method in our subsequent experiments with AC-SUM-GAN. CSNet[8] and SUM-GAN-AAE[2] do not have an unsupervised model selection method. Since their training strategies are similar to that of CA-SUM, following the model selection method of CA-SUM, we select the best model unsupervisedly for SUM-GAN-AAE and CSNet by choosing the best model with minimum model loss on the validation set ($L_{model} = L_{sparsity} + L_{recon}$ for SUM-GAN-AAE and $L_{model} = L_{recon} + L_{prior} + L_{sparsity} + L_v$ for CSNet). Meanwhile, to better assess a model’s performance, we run each model five times on each dataset with different random seeds and report the average.

4.1 Datasets

We evaluate the performance of the model on two public datasets and four datasets we created. The two public datasets are SumMe, Gygli et al.[7], and TVSum, Song et al.[13]. The four datasets we created are Soccer, LoL, MLB, and ShortMLB, where each video is labeled by only one summary.

- **SumMe**: It contains 25 videos with diverse contents (e.g., scuba diving, cooking, cockpit landing) from 1 minute to 6 minutes, captured from both moving and static views. Each video has been annotated by 15 to 18 keyframe fragments generated by different human evaluators. The average summary length is from 10.7% to 15.5% of the original video length. There are 20 training videos and 5 testing videos as in previous approaches[2, 5, 1, 3, 4].
- **TVSum**: It includes 50 videos of various types, such as news, vlogs, and documentaries, with lengths ranging from 1 to 11 minutes. Each video has been evaluated by 20 different human evaluators, who assigned a score of 1 to 5, with 1 indicating not important and 5 indicating very important, to every 2-second shot in the video. We use 40 videos for training and 10 videos for testing according to previous works[2, 5, 1, 3, 4].
- **Soccer**: It consists of 69 videos clipped from 11 soccer games where the video length ranges from 2 to 11 minutes. Nine videos are in the test set, and the other 60 videos are split into 50 training videos and 10 validation videos. Each video in the test set has a goal, which we label as a ground-truth summary.
- **LoL**: It comprises 55 videos extracted from 19 League of Legends matches, with lengths varying from 2 to 10 minutes. Out of these, 5 videos are designated for testing, while the remaining 50 videos are divided into 40 for training and 10 for validation. The ground-truth summary for each video in the test set is composed of segments related to the killing of a hero or the destruction of a tower.
- **MLB**: It has 60 videos from 5 MLB games, with durations between 5 and 10 minutes. Ten of these videos are selected for testing, and the remaining 50 are divided into 40

for training and 10 for validation. The ground-truth summary for each video in the test set is determined by frames that corresponding to a hit.

- **ShortMLB**: This dataset is a shorter version of MLB. We create ShortMLB by clipping each video in MLB to only 2 to 4 minutes. Thus, except for video length, the rest of this dataset is the same as MLB.

We create five random train-test splits for TVSum and SumMe following previous approaches[5, 12, 2, 1], and five random train-test splits for Soccer, LoL, MLB, and ShortMLB.

4.2 Evaluation

Following previous approaches[5, 2, 1, 4], we calculate the F-score to evaluate the quality of the summary generated by the model.

For a single video, we compare the model-generated summary with user-generated summaries (for TVSum and SumMe) or the ground-truth summary (for Soccer, LoL, MLB, and ShortMLB) by computing the F-score for each pair of compared summaries. This F-score is the final F-score for this video for Soccer, LoL, MLB, and ShortMLB datasets. Each video in TVSum and SumMe has multiple user-generated summaries and thus has multiple F-scores. According to the study of SumMe and TVSum by Apostolidis et al.[5], there is no ideal summary that exhibits significant overlap with all annotators’ preferences in SumMe. Moreover, based on the consistency analysis for SumMe and TVSum by Gygli et al.[7] and Song et al.[13], user-generated summaries in TVSum are more consistent for a single video than those in SumMe. Therefore, following the evaluation criteria in previous research[5, 2, 1, 4], we take the maximum of the multiple F-scores to assess the model performance for SumMe and the average of the multiple F-scores for TVSum. We report the average performance over all splits for each dataset.

4.3 Implementation Details

Following previous literature[12, 5, 2, 1, 4], we subsample each video to 2fps and embed each frame to a vector of size $d = 1024$ using GoogLeNet introduced by Szegedy et al.[14] and trained on the ImageNet dataset. We set the regularization factor $\sigma = 0.7$, the temperature $\tau = 0.5$, and the summary rate $\alpha = 0.15$. All bidirectional LSTMs in the model have two layers with the hidden dimension $d_h = 512$. The linear layer in the selector has the input dimension $d = 1024$ and output dimension $d_h = 512$. During training with Adam, we set the learning rate to 0.0001 and the gradient clipping range to $[-5, 5]$. We initialize the model weights randomly. In one iteration, we first train the reconstructor for 100 epochs and then train the selector for 100 epochs.

We propose five versions of the proposed model. Each version has a unique training strategy as follows.

- **SUM-SR**: We train the proposed model for 100 epochs without separating the reconstructor and the selector. The mask vector m is a constant zero vector. This corresponds to one iteration in Figure 2 with training the reconstructor and selector together.

- **SUM-SR_{sep}**: We train the reconstructor and the selector separately for 100 epochs but leave m as a zero vector. This corresponds to one iteration in following iterations in Figure 2.
- **SUM-SR_{sepMa}**: We first train the mask vector m with the reconstructor for 100 epochs, followed by training the selector only for 100 epochs. This corresponds to the first iteration with one-shot training in Figure 2.
- **SUM-SR_{sep-Ma}**: We separate the training of the mask vector m from the training of the reconstructor. We first train the mask vector for 100 epochs. Then, we train the reconstructor for 100 epochs. Finally, we train the selector for 100 epochs. This corresponds to the first iteration with "mask + reconstructor" training in Figure 2.
- **SUM-SR_{5iter}**: We apply the iterative training strategy to SUM-SR_{sepMa} for five iterations. We only update the mask vector m in the first iteration. This corresponds to the entire training part in Figure 2.

We train on NVIDIA GPU cards A100-PCIE-40GB GPU, GeForce GTX 1080, and GeForce RTX 2080 Ti. We used PyTorch version 1.0.1 with Python 3.6 as the development framework.

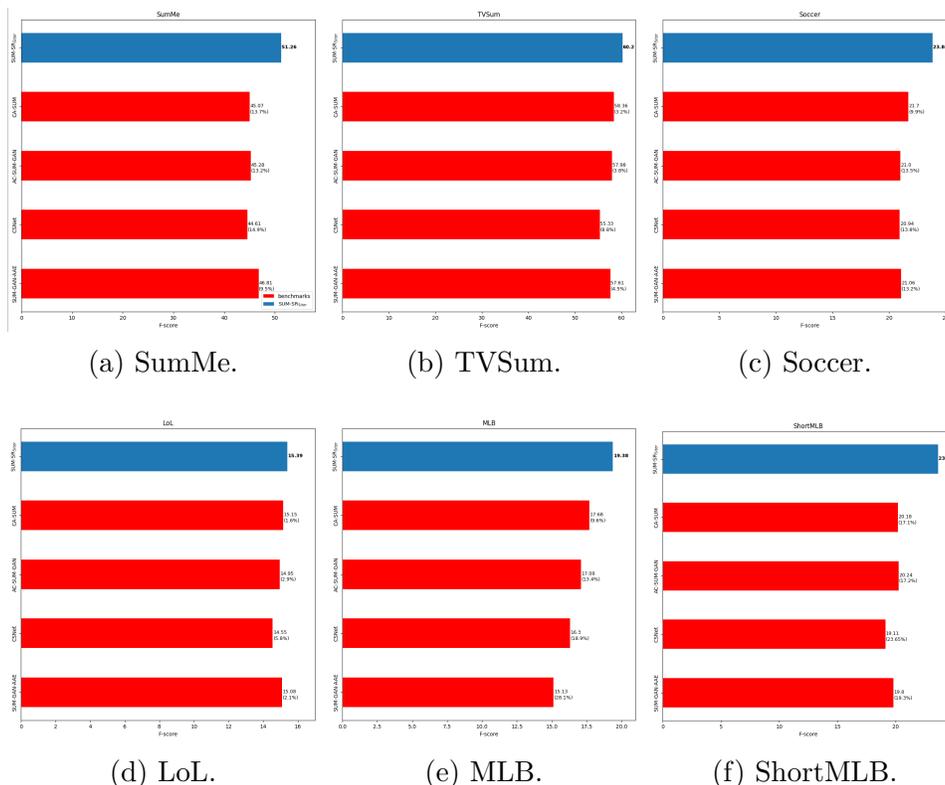


Figure 4: Comparison (F-score (%)) of the proposed approach and state-of-the-art methods of unsupervised video summarization. Between the parentheses below each F-score, we also report the SUM-SR_{5iter}'s relative improvement (in percentage) on each dataset compared to the underlying method.

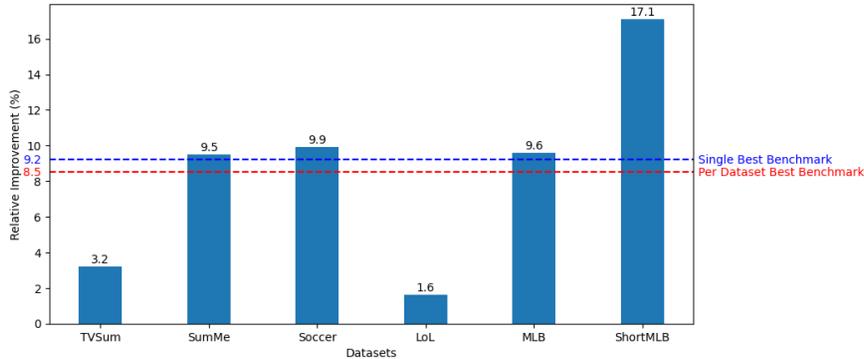


Figure 5: Relative improvement of SUM-SR_{5iter} against best benchmark on the per dataset basis. We label the average improvement based on the per dataset best benchmark and the single best benchmark CA-SUM.

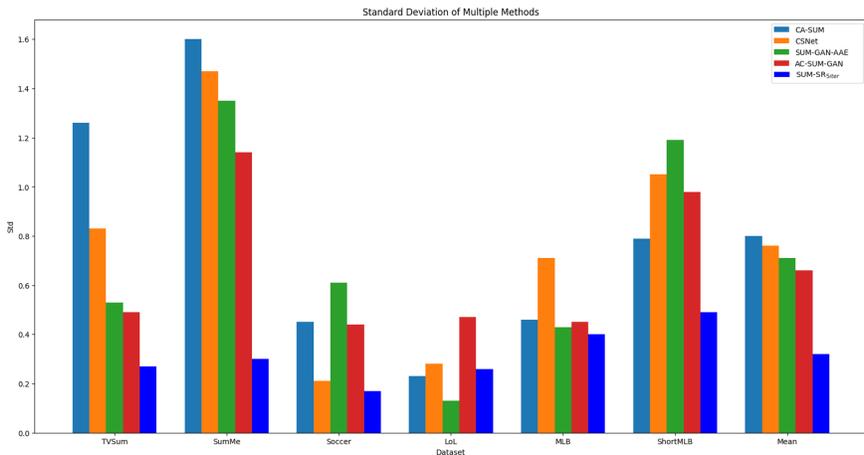


Figure 6: Comparison (standard deviation of the F-score) of different methods running multiple times with different random seeds on six datasets.

4.4 Results

We compare SUM-SR_{5iter}, our best performer, with state-of-the-art unsupervised video summarization approaches. The results in Figure 4 indicate that SUM-SR_{5iter} performs the best on all datasets, outperforming the best benchmark model CA-SUM by 9.2% on average and per dataset best benchmark by 8.5% (see also Figure 5). The values on SumMe and TVSum of the benchmarks are aligned with those reported in the corresponding papers. The proposed training strategy effectively improves the model’s summarization ability. Moreover, compared to other GAN-based methods, removing the discriminator has minimal effect on model performance.

Since we run each method with different random seeds several times, we investigate each model’s stability by computing the final F-score’s standard deviation over the different seeds. According to Figure 6, SUM-SR_{5iter} has the smallest standard deviation compared to other

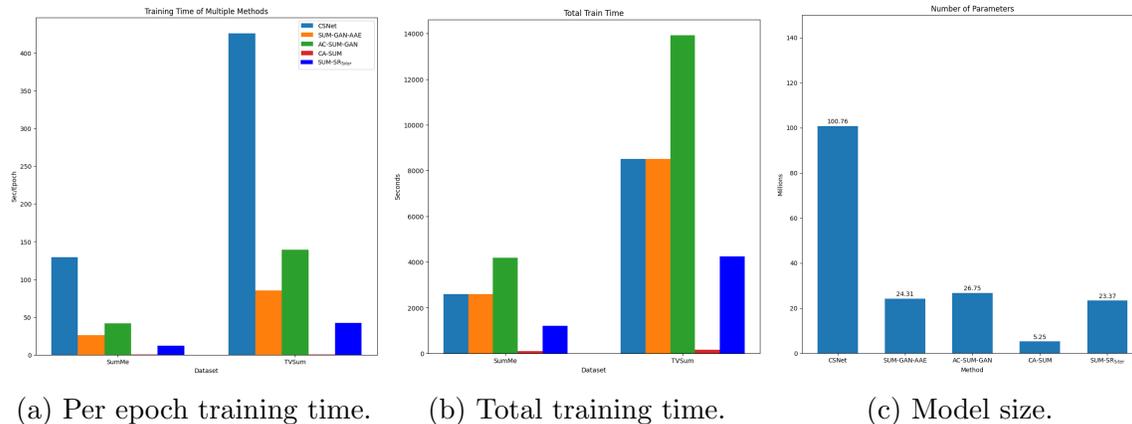


Figure 7: Comparison of per epoch training time (sec/epoch), total training time (seconds) and number of parameters (millions) of different methods in the same computing environment.

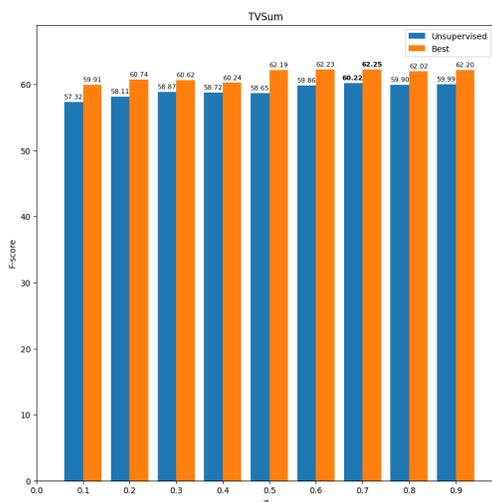


Figure 8: Comparison (F-score (%)) of different σ in SUM-SR_{sepMa} on the TVSum dataset with both unsupervised and supervised (best) model selection methods.

is smaller in size than other GAN-based models and trains much faster except CA-SUM. Eliminating the discriminator simplifies the training

methods on all except one dataset. Although SUM-GAN-AAE and CA-SUM have slightly smaller standard deviations on LoL, they have much higher standard deviations than the proposed model on other datasets. Comparatively, SUM-SR_{5iter} is more resilient to randomness in training.

Moreover, we compare the proposed approach with other unsupervised methods concerning the model size and the training time on SumMe and TVSum. Since different models train for different number of epochs (CSNet for 20 epochs, SUM-GAN-AAE and AC-SUM-GAN for 100 epochs and CA-SUM for 400 epochs), we calculate and compare the per epoch training time of different methods. We run each model in the same computing environment A100-PCIE-40GB over the same five data splits. The results in Figure 7 show that SUM-SR_{5iter}

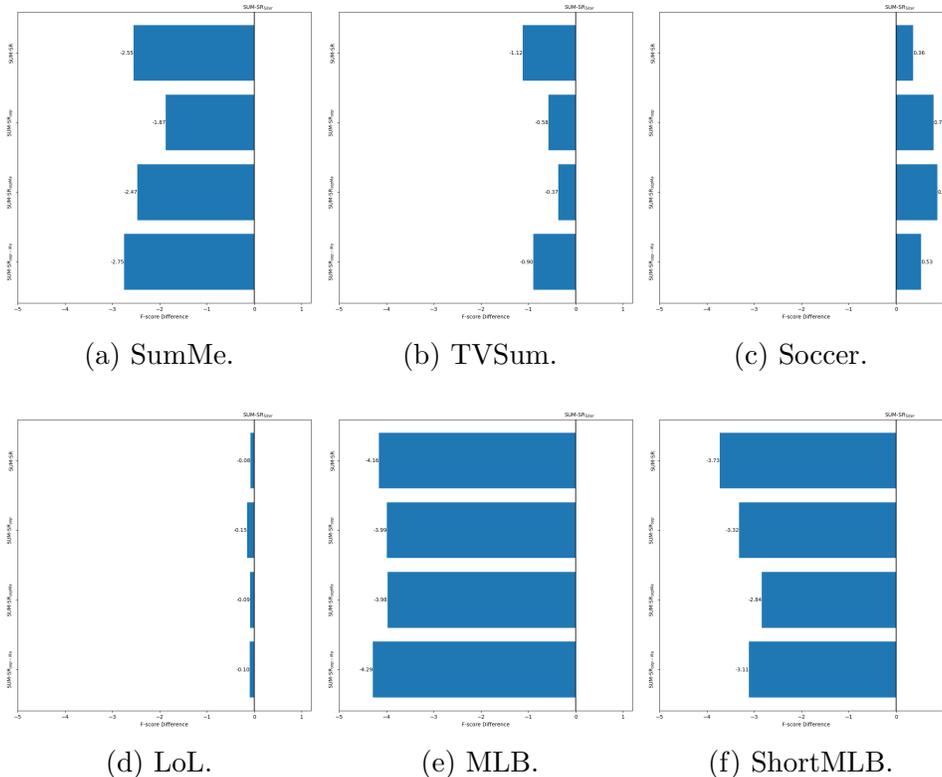


Figure 9: Absolute difference F-score (%) difference between SUM-SR_{5iter} and other variations of the model with no iteration on the six datasets.

step and improves the training efficiency without a performance drop. On the other hand, even though CA-SUM is smaller and trains faster, its performance is worse by 9.2% on average and more unstable than the proposed method.

We also calculate the overall run time by multiplying the per epoch training time with the number of training epochs in Figure 7b. Compared to other GAN-based methods, SUM-SR_{5iter} requires notably less training time. Removing the discriminator decreases the total training time significantly.

4.5 Ablation and Sensitivity Studies

We run sensitivity of the regularization hyperparameter σ and the model versions. To explore the effect of σ , we run one version of the model (SUM-SR_{sepMa}) on TVSum with different σ values from 0.1 to 0.9 and report both the unsupervisedly selected model and the best model. The best model is the model with best performance on test. According to Figure 8, the best option of σ is 0.7. The increment of the σ value does not always lead to performance improvement, but it is evident that models with high σ values (0.6, 0.7, 0.8, 0.9) have better performance than those with low σ values (0.1, 0.2, 0.3, 0.4). There is also

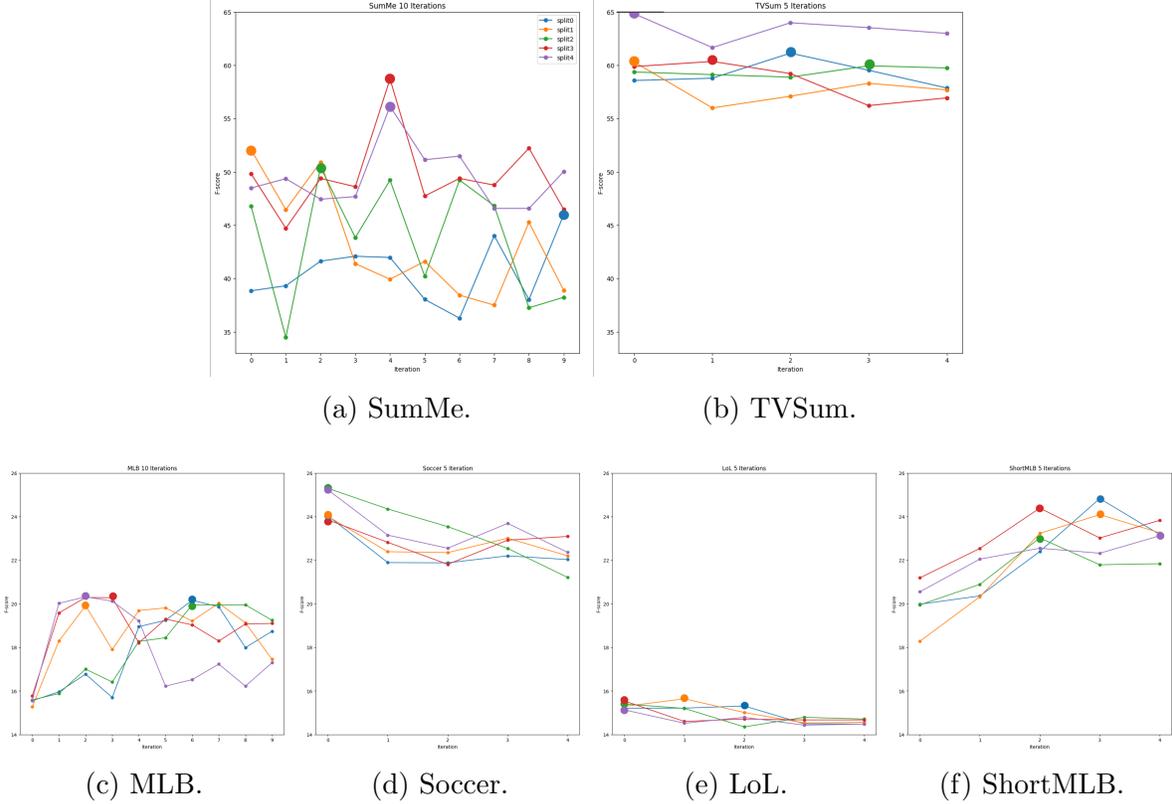


Figure 10: SUM-SR_{sepMa} performance (F-score (%)) over multiple iterations on six datasets. In this figure, we display the result of five splits in one experiment for each dataset. We run SUM-SR_{sepMa} for 10 iterations on SumMe (a) and MLB (c). We run SUM-SR_{sepMa} for 5 iterations on other datasets.

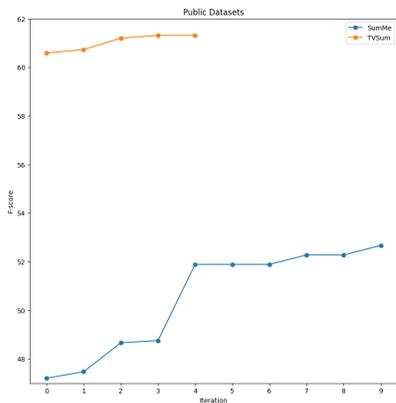


Figure 11: SUM-SR_{sepMa}'s best performance (F-score (%)) over multiple iterations on public datasets SumMe and TVSum.

a sudden increment in the F-score when σ changes from 0.5 to 0.6 with the unsupervised selection method. The difference between the two extreme σ is not remarkable attesting that the model is robust with respect to σ , yet it is beneficial to tune it.

According to Figure 9, SUM-SR_{sepMa} outperforms other versions without iteration on most datasets except SumMe and LoL. On these two datasets, SUM-SR_{sepMa} is the second-best model among variations without iteration. The performance of SUM-SR_{sepMa} suggests that separating the training of the model and using a trainable mask vector m positively affect model performance. However, isolating the training of the mask vector m does not help the model to perform better. Thus, we test the iterative training strategy on SUM-SR_{sepMa}. The results show that the iterative strategy (SUM-SR_{5iter}) further improves the performance of SUM-SR_{sepMa} on five out of six datasets but

degrades the performance slightly on Soccer. According to Figure 10d, SUM-SR_{sepMa} always performs best in the first iteration on Soccer, suggesting that the model in iteration 2 to 5 overfits the training set. Although, our model selection method can select the best iteration at most of the time, a few failures to select the best iteration causing SUM-SR_{5iter} performs slightly worse than SUM-SR_{sepMa} on Soccer. Overall, SUM-SR_{5iter} is the best version of the proposed method.

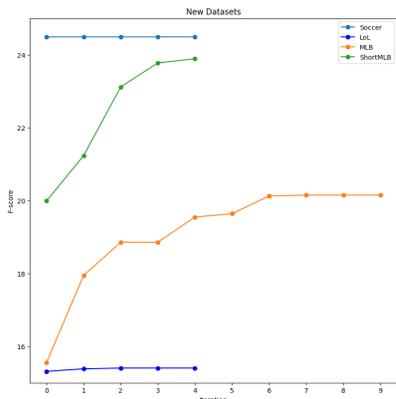


Figure 12: SUM-SR_{sepMa}'s best performance (F-score (%)) over multiple iterations on datasets we created (Soccer, LoL, MLB, and ShortMLB).

For LoL and Soccer, the F-score drops after iteration 0 for most splits (three splits of LoL and all splits of Soccer).

To better evaluate the contribution of the iterative training strategy, we select the highest F-score until the current iteration for each split and average them over five splits. The results are included in Figures 11 and 12 as line charts. The more the iterative strategy improves the model performance, the more the curve increases. According to Figures 11 and 12, the iterative training strategy improves the model performance slightly on TVSum and remarkably on SumMe, MLB, and ShortMLB. For Soccer and LoL, the iterative training strategy has a minimal positive effect on the model performance, which explains the unsatisfactory performance of SUM-SR_{5iter} on Soccer and LoL compared to other versions in Figure 9. In general, the iterative strategy improves the model performance on most datasets. Since the unsupervised model selection method can pick the best iteration most of the time, the model with iterative training performs well on all datasets.

To examine the discrepancy between the summary generated by the model and the reference summary, we analyze two test videos selected from TVSum and MLB. The visual representations of these cases are illustrated in Figure 13. In Figure 13a, the model's choice encompasses two consecutive shots featuring a person riding a bicycle haphazardly, whereas the ground truth summary comprises of two other shots depicting a person positioned on the curb and a host demonstrating the wheel. In Figure 13b, the model-generated summary includes a brief clip of the moving team logo "B", which is absent in the ground truth summary. Additionally, it selects a shot of the camera tracing the ball and infielders fielding

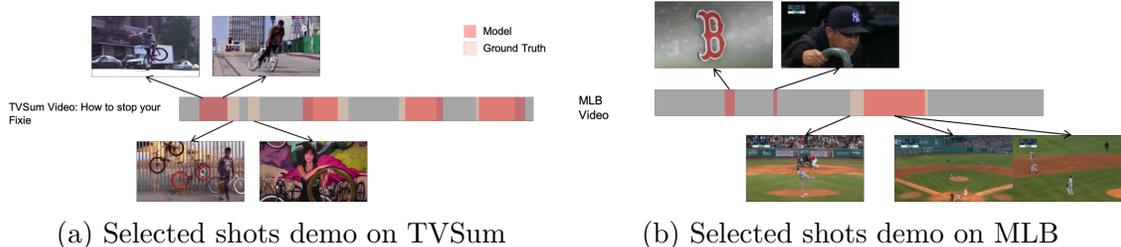


Figure 13: Sample demonstrations of model generated summaries and user-generated summaries.

a ground ball, yet it omits capturing the hit at the previous at bat. Based on these observations, the model prefers video shots with moving objects and a dynamic camera view. Also, the model prefers video shots that are distinct from the overall content of the video. In Figure 13b, the model picks a shot of the team logo in motion (distinct property) and a shot of a manager in a dugout (also distinct due to his dark uniform), which have no similar content in the rest of the video.

5 Conclusion

We present a video summarization model that utilizes an autoencoder for unsupervised training and a part-by-part training strategy for performance improvement. Building on SUM-GAN-AAE, Apostolidis et al.[2], we create a variation that removes the discriminator and separates the training of the selector and the reconstructor. We also explore the iterative training method that trains the model with multiple iterations. Experiments on two public datasets (SumMe and TVSum) and four datasets of ourselves (Soccer, LoL, MLB, ShortMLB) show that removing the discriminator does not impair the model performance but decreases the model size and the training time. The proposed training strategy notably improves the model performance and makes the model outperform the best state-of-the-art method by 9.2% and on per dataset best benchmark by 8.5% on average.

References

- [1] Evlampios Apostolidis, Eleni Adamantidou, Alexandros I Metsai, Vasileios Mezaris, and Ioannis Patras. “AC-SUM-GAN: Connecting Actor-Critic and Generative Adversarial Networks for Unsupervised Video Summarization”. In: *IEEE Transactions on Circuits and Systems for Video Technology*. Vol. 31. 8. IEEE, 2020, pp. 3278–3292.
- [2] Evlampios Apostolidis, Eleni Adamantidou, Alexandros I Metsai, Vasileios Mezaris, and Ioannis Patras. “Unsupervised Video Summarization via Attention-Driven Adversarial Learning”. In: *MultiMedia Modeling: 26th International Conference*. 2020, pp. 492–504.
- [3] Evlampios Apostolidis, Eleni Adamantidou, Alexandros I Metsai, Vasileios Mezaris, and Ioannis Patras. “Video Summarization Using Deep Neural Networks: A Survey”. In: *Proceedings of the IEEE*. Vol. 109. 11. IEEE, 2021, pp. 1838–1863.

- [4] 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*. 2022, pp. 407–415.
- [5] Evlampios Apostolidis, Alexandros I Metsai, Eleni Adamantidou, Vasileios Mezaris, and Ioannis Patras. “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.
- [6] Neeharika Gonuguntla, Bappaditya Mandal, and Niladri B. Puhon. “Enhanced Deep Video Summarization Network”. In: *30th British Machine Vision Conference*. 2019, pp. 1–9.
- [7] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. “Creating Summaries From User Videos”. In: *Computer Vision–ECCV 2014: 13th European Conference*. 2014, pp. 505–520.
- [8] 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*. Vol. 33. 01. 2019, pp. 8537–8544.
- [9] Yunjae Jung, Donghyeon Cho, Sanghyun Woo, and In So Kweon. “Global-and-Local Relative Position Embedding for Unsupervised Video Summarization”. In: *Computer Vision–ECCV 2020: 16th European Conference*. 2020, pp. 167–183.
- [10] Hussain Kanafani, Junaid Ahmed Ghauri, Sherzod Hakimov, and Ralph Ewerth. “Unsupervised Video Summarization via Multi-Source Features”. In: *Proceedings of the 2021 International Conference on Multimedia Retrieval*. 2021, pp. 466–470.
- [11] Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. “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.
- [12] 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*. 2017, pp. 202–211.
- [13] 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*. 2015, pp. 5179–5187.
- [14] 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*. 2015, pp. 1–9.

- [15] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. “Temporal Segment Networks: Towards Good Practices for Deep Action Recognition”. In: *Computer Vision–ECCV 2016: 14th European Conference*. 2016, pp. 20–36.
- [16] Gokhan Yaliniz and Nazli Ikizler-Cinbis. “Using Independently Recurrent Networks for Reinforcement Learning Based Unsupervised Video Summarization”. In: *Multimedia Tools and Applications*. Vol. 80. 12. 2021, pp. 17827–17847.
- [17] Bin Zhao, Xuelong Li, and Xiaoqiang Lu. “Property-Constrained Dual Learning for Video Summarization”. In: *IEEE Transactions on Neural Networks and Learning Systems*. Vol. 31. 10. IEEE, 2019, pp. 3989–4000.
- [18] 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*. Vol. 32. 1. 2018, pp. 7582–7589.