Unsupervised Video Summarization via Iterative Training and Simplified GAN

Hanqing Li[1[0009−0003−5354−7675]], Diego Klabjan[1[0000−0003−4213−9281]], and Jean Utke[2[0000−0002−3377−1990]]

[1] Northwestern University, Evanston IL 60208, USA
{hanqingli2025@u.,d-klabjan@}northwestern.edu
[2] Allstate Insurance Company, Northbrook IL, 60062, USA
jutke@allstate.com

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 Short-MLB) 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.

Keywords: Video summarization · Unsupervised learning · Iterative learning.

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 SUM-GAN-AAE while removing the discriminator. SUM-SR consists of 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 using an additional discriminator to compare the original video with a summary-based reconstructed version like other works [1, 2, 4, 5, 7, 10–12, 16], 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.* [20], and SumMe, Gygli *et al.* [8], 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. The implementation and datasets are available at https://github.com/hanklee97121/SUM-SR-5iter/tree/main.

Our contributions are as follows.

- We create a new framework for the task of unsupervised video summarization by comparing a summary-based reconstructed video 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 Sec. 2, we review previous research on unsupervised video summarization. In Sec. 3, we detail the proposed unsupervised deep learning approach. In Sec. 4, we present the experimental results and compare them to state-of-the-art methods. Finally, in Sec. 5, we conclude the paper.

2 Related Work

In recent years, there have been several approaches to automatic video summarization and related fields such as video highlight detection and video sementic compression. Highlight detection aims to extract brief video segments from unedited recordings that capture the user's primary focus or interest. Supervised approaches [9, 21, 26] predict fine-grained highlight scores, while unsupervised

methods [6,13] identify highlight segments without human annotation. Recently, some studies [15,18,28] have also employed text queries to locate desired highlight clips. In contrast to highlight detection, our work focuses on video summarization that generates a comprehensive summary of a video. Another closely related research area is video semantic compression. It focuses on reducing the size of digital video data while preserving essential semantic information necessary for downstream video analysis tasks [24]. Tian *et al.* [24] are the first to introduce this concept, proposing an unsupervised framework for video semantic compression and a special framework tailored for low-bitrate videos [23]. In contrast to video semantic compression, our research concentrates on generating concise video summaries for human comprehension rather than video analysis tasks. There are supervised and unsupervised video summarization methods based on the presence of ground-truth labels. In this section, we focus on presenting relevant papers on unsupervised approaches. If readers are interested in supervised video summarization, they are referred to the review by Apostolidis *et al.* [3].

In unsupervised video summarization, the absence of ground-truth labels is addressed by focusing on key characteristics of effective summaries. Recent methods aim to create summaries that accurately represent the original content. These approaches typically employ Generator-Discriminator architectures and adversarial training to ensure the summarization component produces a summary that can reconstruct the original video effectively [5,11,12]. Mahasseni *et al.* [16] introduced adversarial learning in video summarization by combining a Variational Auto-Encoder, a discriminator, and an LSTM-based keyframe selector. In SUM-GAN-AAE [2], Apostolidis *et al.* replaced the Variational Auto-Encoder with a deterministic attention-based Auto-Encoder, while in AC-SUM-GAN [1], they embedded an Actor-Critic model to merge adversarial and reinforcement learning. In their latest work [4], Apostolidis *et al.* substituted the GAN with an attention mechanism that leverages frame uniqueness and diversity. Jung *et al.* [10], building on Mahasseni *et al.*'s model [16], developed the Chunk and Stride Network (CSNet), which used both local and global temporal information and introduced a variance loss to highlight dynamic scenes.

Some other unsupervised video summarization methods use hand-crafted reward functions to quantify characteristics like representativeness and diversity, employing reinforcement learning for training [7, 7]. Zhou *et al.* [31] used an LSTM-based architecture with rewards for diversity and representativeness, treating summarization as a sequential decision-making process. Zhao *et al.* [30] combined summarization and reconstruction, using reconstruction to assess how well the summary infers the original video. Yaliniz *et al.* [27] applied independent recurrent neural networks [14] with rewards for representativeness, diversity, and temporal coherence.

Compared to GAN-based methods mentioned above [1, 2, 5, 10–12, 16], 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.* [31]. 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 [7, 14, 25, 27, 30, 31] 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 [1, 2, 5, 7, 10–12, 14, 16, 25, 27, 30, 31] 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

Following the problem setting in SUM-GAN [16], we subsample each video and use a pre-trained CNN to encode each frame. Each video is represented by a sequence of vectors $V = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ 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 Fig. 1), analogous to the selector and the reconstructor in SUM-GAN-AAE [2]. 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 $\boldsymbol{h}_i$ of the LSTM to an importance score p_i. The output layer first maps $\boldsymbol{h}_i$ to a two-dimensional vector $\hat{\boldsymbol{h}}_i$ and computes p_i by softmax function with temperature τ as follows:

$$\boldsymbol{h}_i = \text{biLSTM}(\text{Lin}(\boldsymbol{x}_i), \boldsymbol{h}_{i-1}) \tag{1}$$

$$\hat{\boldsymbol{h}}_i = \text{Lin}(\boldsymbol{h}_i) \tag{2}$$

$$p_i = \text{softmax}(\hat{\boldsymbol{h}}_i, \tau)_1. \tag{3}$$

Then we define $S = (p_1, p_2, p_3, ..., p_n)$ 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 $A = (a_1, a_2, ..., a_n)$ 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 $SU = (\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots, \boldsymbol{s}_n)$, where $\boldsymbol{s}_i = \boldsymbol{x}_i$ if $a_i = 1$ and $\boldsymbol{s}_i = \boldsymbol{m}$ if $a_i = 0$. Vector $\boldsymbol{m}$ is a mask vector with dimension d. We explain more about f and $\boldsymbol{m}$ in later sections.

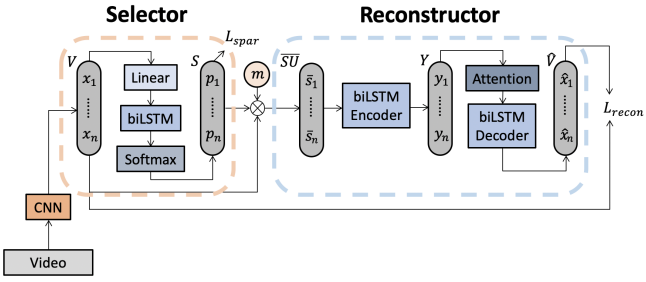


Fig. 1: The proposed SUM-SR architecture.

Since the function f is not differentiable, during training, we use a different method to create a trainable summary $\overline{SU} = (\bar{\boldsymbol{s}}_1, \bar{\boldsymbol{s}}_2, \ldots, \bar{\boldsymbol{s}}_n)$ as in the paper by Mahasseni *et al.* [16]. Each entry $\bar{\boldsymbol{s}}_i$ is a weighted sum of $\boldsymbol{x}_i$ and $\boldsymbol{m}$ given by $\bar{\boldsymbol{s}}_i = p_i \cdot \boldsymbol{x}_i + (1 - p_i) \cdot \boldsymbol{m}$.

The reconstructor of the model is an autoencoder with an attention block introduced in the work by Apostolidis *et al.* [2]. 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 $Y = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n)$, where $\boldsymbol{y}_i \in \mathbb{R}^{d_h}$, and the previous hidden state of the decoder, $\boldsymbol{z}_{i-1} \in \mathbb{R}^{d_h}$. We compute the attention energy vector $\boldsymbol{e}_i \in \mathbb{R}^n$ from Y and $\boldsymbol{z}_{i-1}$ by $\boldsymbol{e}_i = Y^T W_b \boldsymbol{z}_{i-1}$ where W_b is a trainable matrix. At time step $i = 1$, we use the last hidden state of the encoder $\boldsymbol{h}_e$ to calculate $\boldsymbol{e}_1$. Afterward, we apply a softmax function on $\boldsymbol{e}_i$ to get a normalized attention weight vector $\boldsymbol{w}_i = \text{softmax}(\boldsymbol{e}_i)$ and multiply $\boldsymbol{w}_i \in \mathbb{R}^n$ with the encoder's output to produce a context vector $\boldsymbol{y'}_i = Y\boldsymbol{w}_i$. The context vector $\boldsymbol{y'}_i \in \mathbb{R}^{d_h}$ 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 $\hat{V} = (\hat{\boldsymbol{x}}_1, \ldots, \hat{\boldsymbol{x}}_n)$, $\hat{\boldsymbol{x}}_i \in \mathbb{R}^d$, as follows:

$$Y = Encoder(\overline{SU}) \tag{4}$$

$$\hat{V} = Decoder_{atten}(Y). \tag{5}$$

$Decoder_{atten}$ is a bi-directional LSTM with the attention block m_attn.

During training, we use two loss functions introduced in SUM-GAN [16]: 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.* [16], which penalizes the model when it assigns high importance scores to a large number of frames and introduces diversity in the video summary. 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 Fig. 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 $\boldsymbol{x}_i$ from a video $V = (\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)$ with a mask vector $\boldsymbol{m}$ 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 $\boldsymbol{m}$ to get $V' = (\boldsymbol{x'}_1, \boldsymbol{x'}_2, \ldots, \boldsymbol{x'}_n)$, where $p(\boldsymbol{x'}_i = \boldsymbol{x}_i) = \alpha$, $p(\boldsymbol{x'}_i = \boldsymbol{m}) = 1-\alpha$. We feed V' into the reconstructor to get a reconstructed video $\hat{V}' = (\hat{\boldsymbol{x}}'_1, \hat{\boldsymbol{x}}'_2, \ldots, \hat{\boldsymbol{x}}'_n)$, and train the reconstructor by $L_{recon} = ||V - \hat{V}'||^2$. Then, we train only the selector based on L_{model} in Sec. 3.1.

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

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 the paper by Potapov *et al.* [19]. 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 in the work by Gygli *et al.* [8]

$$\max_{\hat{A}} \sum_{i=1}^{N} \hat{a}_i \cdot \hat{p}_i, \text{ s.t.} \sum_{i=1}^{N} \hat{a}_i \cdot l_i \leq \alpha \cdot L, \hat{a}_i \in \{0, 1\}, \tag{6}$$

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 $\hat{A} = (\hat{a}_1, \hat{a}_2, ..., \hat{a}_N)$.

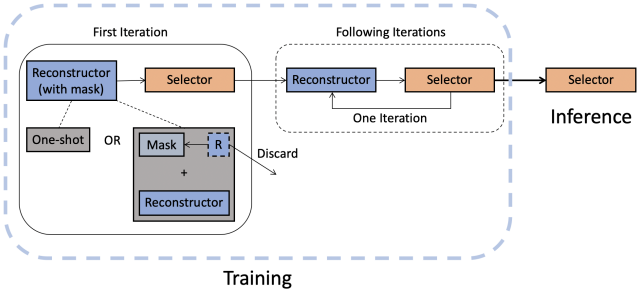


Fig. 2: The training steps of SUM-SR. During training, an iteration includes one reconstruction and one selection. We train the mask vector only in the first iteration (if there are multiple iterations).

3.4 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 Sec. 3.1 and obtain the reconstructed video using the reconstructor ($rNet_i$). Here $i = 1, ..., E$

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 Sec. 3.1. Note that this is different from training, where $\overline{SU}$ is used because of to the need for differentiability. Here, SU_{ij} is the actual summary we construct from the model. Finally, we get the reconstructed 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 $L_{spar_{ij}}$.

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 $\bar{L}^{normal}_{recon_i}$ and $\bar{L}^{normal}_{spar_i}$. For epoch i, the selected model is

$$\underset{i}{argmax}(\bar{L}^{normal}_{recon_i} - \bar{L}^{normal}_{spar_i}). \tag{7}$$

A more detailed rationale and discussion regarding this selection methodology can be found in Appendix A.

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 [10] and SUM-GAN-AAE [2] as benchmarks for performance comparison. According to previous works [1, 4, 10], 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 [10] 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 with an unsupervised method 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.* [8], and TVSum, Song *et al.* [20]. 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 [1–5].
- **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 [1–5].
- **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 [1, 2, 5, 16], and five random train-test splits for Soccer, LoL, MLB, and ShortMLB.

4.2 Evaluation

Following the previous approach by Zhang *et al.* [29], 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.* [8] and Song *et al.* [20], user-generated summaries in TVSum are more consistent for a single video than those in SumMe. Therefore, following the evaluation criteria in the work by Zhang *et al.* [29], we take the maximum of the multiple F-scores to access 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 the paper by Mahasseni *et al.* [16], we subsample each video to 2fps and embed each frame to a vector of size $d = 1024$ using GoogLeNet introduced by Szegedy *et al.* [22] 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 $\boldsymbol{m}$ is a constant zero vector. This corresponds to one iteration in Fig. 2 with training the reconstructor and selector together.
- **SUM-SR$_{\mathbf{sep}}$**: We train the reconstructor and the selector separately for 100 epochs but leave $\boldsymbol{m}$ as a zero vector. This corresponds to one iteration in following iterations in Fig. 2.
- **SUM-SR$_{\mathbf{sepMa}}$**: We first train the mask vector $\boldsymbol{m}$ 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 Fig. 2.

Table 1: Comparison (F-score (%)) of the proposed approach and state-of-the-art methods of unsupervised video summarization.

Method	SumMe	TVSum	Soccer	LoL	MLB	ShortMLB
SUM-GAN-AAE [2]	46.81	57.61	21.06	15.08	15.13	19.8
CSNet [10]	44.61	55.33	20.94	14.55	16.3	19.11
AC-SUM-GAN [1]	45.28	57.98	21.00	14.95	17.09	20.24
CA-SUM [4]	45.07	58.36	21.70	15.15	17.68	20.18
SUM-SR$_{\mathbf{5iter}}$	**51.26**	**60.2**	**23.84**	**15.39**	**19.38**	**23.63**

Table 2: SUM-SR$_{5iter}$'s relative improvement (in percentage) on each dataset compared to the underlying method. We also calculate the average improvement based on the per dataset best bench- mark and the single best benchmark CA-SUM.

Method	SumMe	TVSum	Soccer	LoL	MLB	ShortMLB	Average
SUM-GAN-AAE [2]	**9.5%**	4.5%	13.2%	2.1%	28.1%	19.3%	12.8%
CSNet [10]	14.9%	8.8%	13.8%	5.8%	18.9%	23.65%	14.3%
AC-SUM-GAN [1]	13.2%	3.8%	13.5%	2.9%	13.4%	17.2%	10.7%
CA-SUM [4]	13.7%	**3.2%**	**9.9%**	**1.6%**	**9.6%**	**17.1%**	**9.2%**
Best	9.5%	3.2%	9.9%	1.6%	9.6%	17.1%	8.5%

- **SUM-SR$_{\mathbf{sep-Ma}}$**: We separate the training of the mask vector $\boldsymbol{m}$ 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 Fig. 2.
- **SUM-SR$_{\mathbf{5iter}}$**: We apply the iterative training strategy to SUM-SR$_{sepMa}$ for five iterations. We only update the mask vector $\boldsymbol{m}$ in the first iteration. This corresponds to the entire training part in Fig. 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.

4.4 Results

We compare SUM-SR$_{\mathbf{5iter}}$, our best performer, with state-of-the-art unsupervised video summarization approaches. The results in Tab. 1 and Tab. 2 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%. 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.

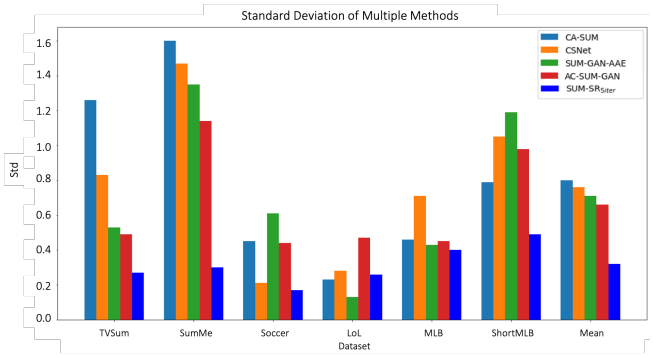
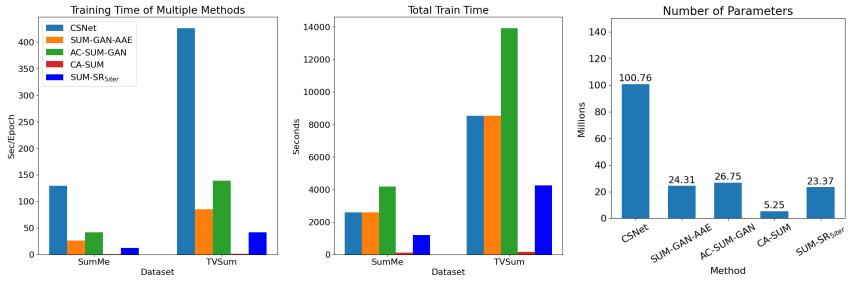


Fig. 3: Comparison (standard deviation of the F-score) of different methods running multiple times with different random seeds on six datasets.



(a) Per epoch training time. **(b)** Total training time. **(c)** Model size.

Fig. 4: 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.

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 Fig. 3, SUM-SR$_{5iter}$ has the smallest standard deviation compared to other 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 Fig. 4 show that SUM-SR$_{5iter}$ is smaller in size than other GAN-based models and trains much faster except CA-SUM. The

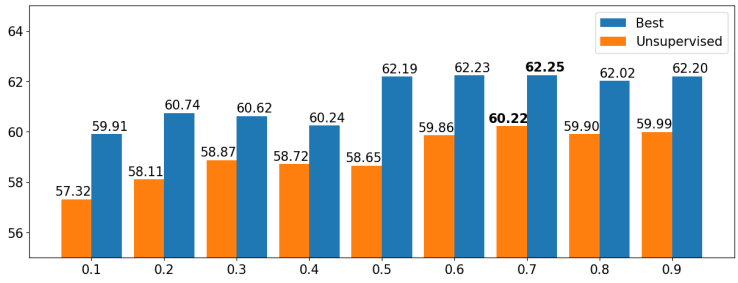


Fig. 5: Comparison (F-score (%)) of different σ in SUM-SR$_{sepMa}$ on the TVSum dataset with both unsupervised and supervised (best) model selection methods.

total training time is 1,203 seconds on SumMe and 4,241 seconds on TVSum. Each training step takes approximately half of the total training time. Eliminating the discriminator simplifies the training 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 Fig. 4b. Compared to other GAN-based methods, SUM-SR$_{5iter}$ requires notably less training time. Removing the discriminator decreases the total training time significantly.

For longer videos (20 minutes to an hour), we conducted experiments on a private dataset; therefore, the results are not included herein. Based on these experiments, we identify two effective approaches for applying our model to longer videos. The first approach is to decrease the frame sampling frequency to 1 fps or lower, as video content often remains consistent over several seconds. The second approach involves dividing the video into multiple shots using shot boundary detection methods, such as those applied in egocentric videos [17]. Each shot is summarized individually, and these summaries are concatenated before applying the summarization model again to produce a comprehensive summary of the entire video.

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 model selected by our method and the best model. The best model is the model with best performance on test. According to Fig. 5, 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 a sudden increment in the F-score when σ changes from 0.5 to 0.6 with the unsupervised

Table 3: Comparison (F-score (%)) of SUM-SR$_{5iter}$ and other variations of the model with no iteration on the six datasets.

Method	SumMe	TVSum	Soccer	LoL	MLB	ShortMLB
SUM-SR	48.71	59.08	24.2	15.31	15.22	19.9
SUM-SR$_{sep}$	49.39	59.62	24.63	15.24	15.39	20.31
SUM-SR$_{sepMa}$	48.79	59.83	**24.71**	15.3	15.4	20.79
SUM-SR$_{sep-Ma}$	48.51	59.3	24.37	15.29	15.09	20.52
SUM-SR$_{\mathbf{5iter}}$	**51.26**	**60.2**	23.84	**15.39**	**19.38**	**23.63**

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 Tab. 3, 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 $\boldsymbol{m}$ positively affect model performance. However, isolating the training of the mask vector $\boldsymbol{m}$ 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. Overall, SUM-SR$_{5iter}$ is the best version of the proposed method. We include further analysis of iterations in Appendix B and examples of video summaries in Appendix C.

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. Apostolidis, E., Adamantidou, E., Metsai, A.I., Mezaris, V., Patras, I.: 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, pp. 3278–3292. IEEE (2020)
2. Apostolidis, E., Adamantidou, E., Metsai, A.I., Mezaris, V., Patras, I.: Unsupervised video summarization via attention-driven adversarial learning. In: MultiMedia Modeling: 26th International Conference. pp. 492–504 (2020)
3. Apostolidis, E., Adamantidou, E., Metsai, A.I., Mezaris, V., Patras, I.: Video summarization using deep neural networks: A survey. In: Proceedings of the IEEE. vol. 109. 11, pp. 1838–1863. IEEE (2021)
4. Apostolidis, E., Balaouras, G., Mezaris, V., Patras, I.: 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. pp. 407–415 (2022)
5. Apostolidis, E., Metsai, A.I., Adamantidou, E., Mezaris, V., Patras, I.: 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. pp. 17–25 (2019)
6. Badamdorj, T., Rochan, M., Wang, Y., Cheng, L.: Contrastive learning for unsupervised video highlight detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14042–14052 (2022)
7. Gonuguntla, N., Mandal, B., Puhan, N.B.: Enhanced deep video summarization network. In: 30th British Machine Vision Conference. pp. 1–9 (2019)
8. Gygli, M., Grabner, H., Riemenschneider, H., Van Gool, L.: Creating summaries from user videos. In: Computer Vision–ECCV 2014: 13th European Conference. pp. 5179–5187 (2014)
9. Gygli, M., Song, Y., Cao, L.: Video2gif: Automatic generation of animated gifs from video. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1001–1009 (2016)
10. Jung, Y., Cho, D., Kim, D., Woo, S., Kweon, I.S.: Discriminative feature learning for unsupervised video summarization. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 33. 01, pp. 8537–8544 (2019)
11. Jung, Y., Cho, D., Woo, S., Kweon, I.S.: Global-and-local relative position embedding for unsupervised video summarization. In: Computer Vision–ECCV 2020: 16th European Conference. pp. 167–183 (2020)
12. Kanafani, H., Ghauri, J.A., Hakimov, S., Ewerth, R.: Unsupervised video summarization via multi-source features. In: Proceedings of the 2021 International Conference on Multimedia Retrieval. pp. 466–470 (2021)
13. Khosla, A., Hamid, R., Lin, C.J., Sundaresan, N.: Large-scale video summarization using web-image priors. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2698–2705 (2013)
14. Li, S., Li, W., Cook, C., Zhu, C., Gao, Y.: Independently recurrent neural network (indrnn): Building a longer and deeper rnn. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5457–5466 (2018)
15. Lin, K.Q., Zhang, P., Chen, J., Pramanick, S., Gao, D., Wang, A.J., Yan, R., Shou, M.Z.: Univtg: Towards unified video-language temporal grounding. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 2794–2804 (October 2023)

16. Mahasseni, B., Lam, M., Todorovic, S.: Unsupervised video summarization with adversarial lstm networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 202–211 (2017)
17. del Molino, A.G., Tan, C., Lim, J.H., Tan, A.H.: Summarization of egocentric videos: A comprehensive survey. In: IEEE Transactions on Human-Machine Systems. vol. 47, pp. 65–76 (2017). https://doi.org/10.1109/THMS.2016.2623480
18. Moon, W., Hyun, S., Park, S., Park, D., Heo, J.P.: Query-dependent video representation for moment retrieval and highlight detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 23023–23033 (June 2023)
19. Potapov, D., Douze, M., Harchaoui, Z., Schmid, C.: Category-specific video summarization. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13. pp. 540–555. Springer (2014)
20. Song, Y., Vallmitjana, J., Stent, A., Jaimes, A.: Tvsum: Summarizing web videos using titles. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5179–5187 (2015)
21. Sun, M., Farhadi, A., Seitz, S.: Ranking domain-specific highlights by analyzing edited videos. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13. pp. 787–802. Springer (2014)
22. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1–9 (2015)
23. Tian, Y., Lu, G., Yan, Y., Zhai, G., Chen, L., Gao, Z.: A coding framework and benchmark towards low-bitrate video understanding. IEEE (2024)
24. Tian, Y., Lu, G., Zhai, G., Gao, Z.: Non-semantics suppressed mask learning for unsupervised video semantic compression. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13610–13622 (2023)
25. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Van Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: Computer Vision–ECCV 2016: 14th European Conference. pp. 20–36 (2016)
26. Xu, M., Wang, H., Ni, B., Zhu, R., Sun, Z., Wang, C.: Cross-category video highlight detection via set-based learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7970–7979 (2021)
27. Yaliniz, G., Ikizler-Cinbis, N.: Using independently recurrent networks for reinforcement learning based unsupervised video summarization. In: Multimedia Tools and Applications. vol. 80. 12, pp. 17827–17847 (2021)
28. Zala, A., Cho, J., Kottur, S., Chen, X., Oguz, B., Mehdad, Y., Bansal, M.: Hierarchical video-moment retrieval and step-captioning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 23056–23065 (June 2023)
29. Zhang, K., Chao, W.L., Sha, F., Grauman, K.: Video summarization with long short-term memory. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14. pp. 766–782. Springer (2016)
30. Zhao, B., Li, X., Lu, X.: Property-constrained dual learning for video summarization. In: IEEE Transactions on Neural Networks and Learning Systems. vol. 31. 10, pp. 3989–4000. IEEE (2019)

31. Zhou, K., Qiao, Y., Xiang, T.: Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32. 1, pp. 7582–7589 (2018)