Comparison between Reward Function with Deep Summarization Network and Actor-Critic with Generative Adversarial Networks for Unsupervised Video Summarization

Truong Minh Chau, Pham Minh Khoi, Le Doan Thien Nhan

Abstract— **This paper introduces the end-to-end DSN framework developed based on reinforcement learning by adding a reward function to the model. Then extend the method to a supervised case to make good use of the labeled data, and finally run experiments to prove the method is better than the published methods. Moreover, the article also provides you a detailed description about a special unsupervised learning architecture that combines a Generative Adversarial Network (GAN) with an Actor-Critic model called AC-SUM-GAN. Later, we will introduce about how to set up the model and using some specific parameters to evaluate the result. Finally, we bring out some figures and make a comparison to the DSN methods and evaluate the partial components within the architecture.**

I. INTRODUCTION

A. Reward Function with DSN

According to the paper by Zhou et al. for video summarization, they proposed to use an unsupervised method. The principal reason why they got here up with this proposal is that supervised learning cannot fully explore the potential of deep networks. Once considering the DPP-LSTM[1] model combined with the supervised method and the use of both video-level summaries and frame-level importance scores by Zhang et al. (2016) to predict importance scores and output feature vectors simultaneously has shown state-of-the-art performances on several benchmarks, albeit a result like that, there is no single ground truth summary for a video by fact. It is based on human subjectivity chosen which part of the video to summarize rather than relying on the label.

Mahasenni et al. (2017) developed an adversarial learning framework to train DPP[2]. All through the training process, their framework is unsupervised, however the antagonistic nature makes the training unstable might also lead to model collapse. In terms of increasing diversity, DPP-LSTM cannot gain maximally from the DPP module without the assist of labels seeing that an encoder-decoder with RNN[3] following DPP-LSTM for video reconstruction calls for pretraining, and their framework calls for multiple training stages. This result is taken into consideration as a supplementary reason that makes Zhou et al. decide to advance another model. They built up a video summarization as a sequential decision-making process and contemporarily developed DSN[4].

The DSN has a decoder and decoder architecture, with the encoder being CNN extracting functions from the video frames and the decoder being a two-way LSTM network generating probabilities from the actions that are sampled to select the frame. An end-to-end reinforcement learning-based framework is proposed with diversity and representative reward function to take into account the diversity and representativeness of the generated summaries, and depend much less on tags or user interactions. The reward for diversity measures how different the selected frames are to each other, whereas the reward for representativeness calculates the distances between the frames and their nearest selected frames, which is basically an algorithm k-medoids. This is the first time that reinforcement learning has been applied to unsupervised video summarization. The goal of DSN learning is to maximize expected rewards over time. The rationale for using RL[5] to train a DSN is two-fold.

Here the RNN is considered as part of the model and unsupervised setting is also focused more. Not only that, but the reward is only received after the completed chain and to ensure the monitoring of the reward at the end of the chain, RL is the most appropriate choice. The author believes that DSN can gain more benefits from RL because this learning method basically aims to optimize the frame-selection mechanism of agent by enforcing agent to take better and better actions. The optimization mechanism is not specially marked in the settings because the training does not require a label. And to accommodate the labeled case, they extend by adding a supervised objective that directly maximizes the log-probability of selecting annotated key-frames.

B. Actor-Critic with GAN

Nowadays, the evolution of multi-media has lead to a tremendous growth of online videos in quantity. In order for people not to waste time watching the whole video, many video summarization methods have been proposed and each of them tackle a specific aspect of the videos. Generally speaking, all those methods are divided into two main groups, supervised method and unsupervised method.

[1]Bidirectional Long Short-Term Memory Network with a Determinantal Point Process

[2]Determinantal Point Process

[3]Recurrent Neural Network

[4]Deep Summarization Network

[5]Reinforcement Learning

Supervised methods base on ground-truth summarizes, those are generated by humans, to find out the primary criterion to summarize videos. However, generating the ground-truth summaries is a tedious and laborious task. Moreover, summarize a video by hand may consequently result in different synopses according to the perspective of each person in selecting the principal information in that video. Some related works using supervised learning can be mentioned as follows.

Early works on video summarization are built on the foundation of CNN/DCNN[6] architectures, some methods generate the summary base on the similarity in semantics between the video itself and some relative videos in the same category. Others using category-driven method to maximize the similarity of the generated summary with other video. However, those aforementioned works have not considered the spatial and temporal characteristic of each frame in the video, which mean the relation of each frame with the prior frames and the later ones.

To overcome the shortcoming of the early supervised video summarization, the later works pay more attention to the temporal structure of the frames in the whole video and the importance of each frame itself. To this direction, some architectures like two-layer LSTM, Sequence-to-Sequence (seq2seq), LSTM-based Encoder-Decoder network, Generator-Discriminator, etc. are used to select key-frames and estimating frames' importance scores.

In a different approach to minimizing the distance between the machine-generate summaries and the ground-truth ones, GAN[7] architectures are used to summarize videos by training the Generator to generates fake summaries to fool the trained Discriminator, whose role is to distinguish whether the summaries are the ground-truth ones or the generated ones. In this type of architecture, not only deep learning is included, but also reinforcement learning. The output of the Discriminator is used to train the Generator to generate realistic summaries that seem to be identical to the human-generated ones.

Contrary to all the above methods, unsupervised learning avoids using ground-truth training data that leads to minimize the human's impact during the training period. These techniques rely on GANs to reconstruct the video from the given summaries, and thus automatically minimizing the distance between the concise synopsis of the video and the video itself. The work of Mahasseni et al. was the first to combines the LSTM-base key-frame selector with VAE[8] and a trainable Discriminator. After that, many variant architectures that based on VAE-GAN like adding a CSN[9] or replacing VAE with an AAE[10] that leads to an improvement in selecting key-fragment of the video.

[6]Convolutional Neural Network/Deep Convolutional Neural Network
[7]Generative Adversarial Network
[8]Variational Auto-Encoder
[9]Chunk and Stride Network
[10]Attention Auto-Encoder

To another direction, some proposes methods that concentrate on some specific properties of the video in order to deal with unstable training and the restricted evaluation criteria of GANs-base methods. Reinforcement learning is brought out to combines with the GANs in training the summarizer to produce diverse and representative summaries using human-crafted reward function.

Inherit the idea from the earlier work in video summarization task, the authors of [3] develop a combination of GAN architecture and reinforcement learning, which uses the Discriminator's feedback to train the summarizer but in a completely unsupervised manner. This model can improve the performance in estimating the importance scores of frames, which exhibits very low variation and so limits it impact in selecting key-fragments to reconstruct the original video. Contrary to the LSTM-based algorithm, this model is embedded with a pair of trainable models Actor and Critic, which automatically explores the space of actions and discovers the underlying criterion of key-fragments of the video by changing the importance score of relevant frames and the irrelevant ones. Therefore, increasing the impact of the selecting fragments task on summarizing the video using Knapsack algorithm. Moreover, the embedded models Critic and Actor automatically learn the values function (Critic) and propose an optimal policy (Actor) in selecting the key-fragment. Thus, it eliminates the need of human being in defining the reward function and makes the model completely unsupervised.

The authors of this architecture inherit the idea from [2], where the similar model is used in NLP[11] tasks, and apply it to CV[12] domain, in which the model takes part in selecting key-fragments of the video to form a concise synopsis of the original video using Knapsack algorithm. Using the same idea from NLP generating tasks, in which the smaller units of the data sequence is called tokens, and they are used to form the data sequence, video summarization is visualized as "visual content" generative process, and the tokens of this process are key-fragments. To materialize the formulation, a GAN architecture, which is embedded with an Actor-Critic model, is proposed and named AC-SUM-GAN.

II. DSN Architecture

As mentioned in the introduction to the video summary model as a sequential decision-making process. The emphasis here is the deep summarization network capable of predicting probabilities for frames, then making a frame selection based on the predicted probability distribution. In addition, the diversity-representativeness reward function is designed within an end-to-end reinforcement learning-based framework model to directly evaluate the diversity and representativeness of abstracts during DSN training.

[11]Natural Language Processing
[12]Computer Vision

A. Deep Summarization Network

The DSN is formulated based on the encoder-decoder version, where the encoder is a CNN that uses GoogLeNet to extract the visual features $x_{t=1}^{T}$ from the input video frames $v_{t=1}^{T}$, after which pass it to a decoder. The decoder backbone is constructed as BiRNN[13] topped with the FC layer[14]. In which, the BiRNN model uses LSTM is responsible for taking all the transferred visual features to create corresponding hidden states $h_{t=1}^{T}$. Each hidden state includes the information of the previous state and the following state, along with emphasis on the parts around the t_{th} frame. Then the hidden states are fed into the FC layer and through the sigmoid function output the probability prediction p_t of each frame, from which the frame-selection action a_t is sampled for a part of the final summary:

$$\begin{aligned} p_t &= \sigma\left(W h_t\right), \\ a_t &\sim \text{Bernoulli}\left(p_t\right) \end{aligned} \tag{1}$$

σ is a symbol of sigmoid function, and W is the learnable parameters. Bernoulli sampling is subsequently implemented to produce either success or failure, so a_t has two values are 0 or 1 to indicate whether the t_{th} frame is picked out or not.

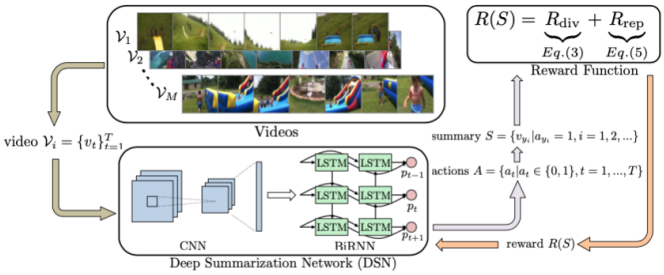


Fig. 1. DSN architecture

B. Reward Function

After completing the frame selection, the next stage goes into the reward function. In this function, the proposed reward will include diversity reward R_{div} and representativeness reward R_{rep}. The reason for creating 2 separate rewards inside the function is because the original goal of the DSN was to maximize the expected reward over time via generating high quality summaries. A high-quality video summary is therefore expected to be both diverse and representative of the original video, so that temporal information across the entire video can be preserved as much as possible. The DSN will then receive feedback for evaluating the quality of the generated summaries as a form of reward.

Diversity reward appraise the diversity of the generated summary via computing the disparate chosen frames in the feature space. The circumstance for the reward to reach a better value if and only if the elected frames are as diverse (or dissimilar) as possible. R_{div} is the formula for calculating the mean of pairwise differences between selected frames:

$$R_{\text{div}} = \frac{1}{|\mathcal{Y}|(|\mathcal{Y}|-1)} \sum_{t \in \mathcal{Y}} \sum_{\substack{t' \in \mathcal{Y} \\ t' \neq t}} d\left(x_t, x_{t'}\right), \tag{2}$$

where $\mathcal{Y} = \{y_i \,|a_{y_i} = 1, i = 1, \ldots, |\,\mathcal{Y}\,|\}$ is the index of chosen shot features and the dissimilarity function $d\left(x_t, x_{t'}\right)$ is estimated by

$$d\left(x_t, x_{t'}\right) = 1 - \frac{x_t^T x_{t'}}{\|x_t\|_2 \|x_{t'}\|_2} \tag{3}$$

However Eq.[3] considers frames to be randomly permutable items, ignoring the inherent temporal structure in sequential data. In practice, such omissions occur during storyline construction, thus to overcome this problem, the researchers added a lambda function, which controls the amount of temporal gap and set $d\left(x_t, x_{t'}\right) = 1$ if $|t - t'| > \lambda$. The addition of this equation improved the model's score.

Representativeness reward is essentially a k-means clustering function, which measures how well the generated summary mirrors the original video. Zhou et al. defined R_{rep} as the lower average of squared errors between video frames and their most adjacent mediods, the higher score:

$$R_{\text{rep}} = \exp\left(-\frac{1}{T} \sum_{t=1}^{T} \min_{t' \in \mathcal{Y}} \|x_t - x_{t'}\|_2\right) \tag{4}$$

Unsupervised reward is a complementary combination of R_{div} and R_{rep} to guide the learning of DSN model:

$$R_{\text{S}} = R_{\text{div}} + R_{\text{rep}} \tag{5}$$

For the training duration, R_{div} and R_{rep} are comparable inside the order of magnitude. In reality, it's miles non-trivial to preserve R_{div} and R_{rep} on the identical order of magnitude at some point of training, for this reason, none of them could dominate in gradient computation. We do not supply rewards to DSN when no frames are selected, i.e., the sampled movements are all zeros.

C. Optimization

The DSN is trained with a REINFORCE algorithm, going to learn a policy function π_θ to maximize the expected rewards:

$$J(\theta) = E_{p_\theta(a_{1:T})}[R(S)] \tag{6}$$

where θ denotes the trainable parameters of the deep summary network and $p_\theta\left(a_{1:T}\right)$ is that of the probability distributions over opening sequences.

Following the REINFORCE algorithm, they will compute the derivative of the objective function $J(\theta)$ with relevance to the parameters θ as

$$\nabla_\theta J(\theta) = E_{p_\theta(a_{1:T})}\left[R \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(a_t \mid h_t\right)\right] \tag{7}$$

[13]Bidirectional Recurrent Neural Network
[14]Fully Connected Layer

where a_t is that of the action taken by time t and h_t is that of the hidden state from the BiRNN.

Eq.[7] involves the expectation over high dimensional action sequences, which is difficult to compute directly, thus we introduce a Monte-Carlo policy gradient method to approximate the gradient by running the agent for N episodes on the identical video for input sequences and so taking the mean gradient. During this time, it is also referred to as the episodic REINFORCE algorithm.

Although the gradient may be a good estimate, it may contain high variance, which is able to make the network hard to converge. A typical countermeasure is to subtract the reward by a constant baseline b, therefore the gradient becomes

$$\nabla_\theta J(\theta) \approx \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T} (R_n - b) \nabla_\theta \log \pi_\theta (a_t \mid h_t) \tag{8}$$

here b is simply the moving average of rewards R_n for computational efficiency.

They optimize the policy function's parameters θ via the stochastic gradient-based method, whilst that of practice, they use Adam algorithm as the optimization one. The main reason is one of Adam's key components is exponential weighted moving averages that use bias correction to adjust for the slow-start case to estimate the momentum and the quadratic moment of the gradient. As a result of learning, the log-probability of actions taken by the network that have a stark contrast results regarding high rewards and low rewards, with the former continuing to witness an upward and the latter takes a nosedive. Formula θ is

$$\theta = \theta - \alpha \nabla_\theta \left(-J + \beta_1 L_{\text{percentage}} + \beta_2 L_{\text{weight}}\right) \tag{9}$$

In Eq.[9], we have α to be a learning rate, while β_1 and β_2 are hyperparameters to balance the weighting. Besides that, $L_{percentage}$ is a regularization term on the probability distributions $p_{1:T}$ to minimize the percentage of frames selected for the summary, as the more picked frames, the more increased reward:

$$L_{\text{percentage}} = \left\| \frac{1}{T} \sum_{t=1}^{T} p_t - \epsilon \right\|^2 \tag{10}$$

with ϵ is the percent chosen frames. Not only adding $L_{percentage}$, but L_{weight} is also another variance of regularization term on weight parameters θ to avoid overfitting:

$$L_{\text{weight}} = \sum_{i,j} \theta_{i,j}^2 \tag{11}$$

D. Extension to Supervised Learning

They expand their model for the supervised case using MLE[15] to maximize the log-probability of selecting key-frames detailed through the key-frame indices. The objective is formalized as

$$L_{\text{MLE}} = \sum_{t \in \mathcal{Y}^*} \log p(t; \theta) \tag{12}$$

[15]Maximum Likelihood Estimation

III. GAN Architecture

A. Overall Architecture

The figure 2 demonstrates the detailed structure of the AC-SUM-GAN model. The sub-figure on the left shows the building blocks of the architecture and the way they connect to each other. The blue blocks are the one that belong to the Actor-Critic model. The sub-figure on the right illustrates the data flow in the AC-SUM-GAN. Moreover, the input and output of each block are also included in the figure, thus, we can understand the role of each part of the architecture. The dashed line implies the iterative parts during the training process of the AC model. All the components bounded with the orange box are those that take part in creating a summary of a new video.

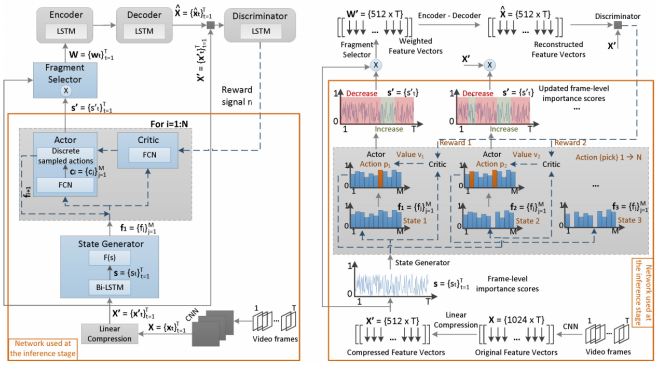


Fig. 2. AC-SUM-GAN architecture

The process of creating a synopsis for a given video start when T frames of that video are fed as input to a CNN-based deep feature network and produces a sequence of frame-level scores that indicates the suitability of each frame in the summary. The following process is passing the deep feature vectors through a fully connected layer that reduce the dimension of these vectors. After that, the State Generator takes those feature vectors to create an initial state of the action-state space of the AC model's training process by assigning each frame a score, which is calculated via an average pooling operation according to the temporal dependency between that frame and the other frames of the video. Having the initial state created, the Actor comes in and plays an "N-picks" game which select N non-overlapping fragments. The result of the previous stage makes some changes in the importance scores, in particular, the scores of those fragments that correspond to the selected ones will be increased, and the others' will be decreased. Finally, the frame-level scores is used to form an output s' which plays a main role in defining the summary with a limited duration (normally 15% of the original video's duration). The

model will calculate the scores using KTS[16] method, which groups similar frames such that semantic changes occur at the boundaries.

A **State Generator** comprises two components. The first part is a bi-directional LSTM. It plays the role in capturing the temporal dependency of the frames in both forward and backward direction and assigns an importance score ($s = \{s_t\}_{t=1}^{T}$ with $s_t \in R$ and $0 \leq s_t \leq 1$) to each of them. The latter part is an average pooling operator, whom role is to produce an initial state f for the AC action-state space from the computed scores s by calculating scores at a coarser fragment-level. To fit the output format of the State Generator, the video must be divided into M non-overlapping fragments and each has a duration of d. The fragment-level scores is computed by averaging the frame-level scores of those frames in the same fragment ($f = \{f_j\}_{j=1}^{M}$ with $f_j \in R$ and $f_j = \frac{\Sigma_{t=(j-1)d+1}^{jd} s_t}{d}$).

An **Actor** is a fully connected network. Its role is to explore the action-state space by iterating N times through the process of selecting the fragments that will be used to form in the summary of the video in the latter stage and at every iteration, the Actor gets the current stage ($f_i = \{f_j\}_{j=1}^{M}$) and produces a distribution of actions $c_i = \{c_j\}_{j=1}^{M}$, then it selects a video fragment k by sampling the distribution signed as action p_i. After that, the process move to the next stage f_{i+1}. In this next stage, the k^th element is assigned with zero ($f_k = 0$) to prevent the Actor from re-selecting this fragment to add into the synopsis. Moreover, the importance scores s of the frames are also affected by the Actor's choices. For instance, the weight of those frames within the selected fragments and those corresponding to them will be increased while reducing the weight of the frames within the fragments that have not been selected at any step of the game using the action-weighting factors. Consequently, the update will result in a new sequence of frame-level weight s'. The action-weighting factors (AwF) at the i^th step are computed as the following formula:

$$AwF_i = \frac{N-(i-1)}{M-(i-1)} + 1, i \in [i, N] \tag{13}$$

The formula indicates that the importance of the former selected fragments means much more than the latter selected fragments, thus the action-weighting factor in step i is higher than the one in step $i+1$.

On the other hand, the weight of frames within the fragments that have not been selected will be updated using the reduction factor (RF), which is computed as follows:

$$RF = \frac{(M-N)}{M} \tag{14}$$

A **Critic** is a fully connected network, who appears in every step i (with $1 \leq i \leq N$) of the selecting video fragments process. Its role is to get the current state f_i, which is either generated by the State Generator at the beginning of the process, or as the output of the Actor at the beginning of every step, and computes the corresponding value v_i, which can be seen as an evaluation of the Actor's actions.

A **Fragment Selector** is a matrix multiplication operator, whose role is to bring the frame-level scores at each step, assign them to the compressed features of the video frames ($X' = \{x'_t\}_{t=1}^{T}$) and create a sequence of the features' weights ($W = \{w_t\}_{t=1}^{T}$).

A **Variational Auto-Encoder** is an LSTM-based generator. The model's function is trying to figure out the weighted data, which is the output of the selecting key fragments process, structure and reconstruct the original video frames ($\hat{X} = \{\hat{x}_t\}_{t=1}^{T}$). This encoding-decoding process aims to optimize the loss function and produce a fake video that fools the Discriminator.

A **Discriminator** is also an LTSM-based model, which forms the AC-environment. In every iteration i ($1 \leq i \leq N$), the model takes the compressed feature vectors of the original video X' and the feature vectors of the reconstructed video $\hat{X}$, then defines a new latent representation for each of the aforementioned versions of the video. Finally, the Discriminator calculates the loss value between the reconstructed video and the original one based on the spatio-temporal compatibility of the frames. The model outputs a reward, which is fed as the input to the Critic later on, computed as follows:

$$r_i = 1 - L_{recon}, r_i \in R, i \in [i, N] \tag{15}$$

When the sampled action of the Actor result in a choice that re-select a video fragment, the output reward of the Discriminator will be zero to penalize the fragment's re-selection.

B. Learning Objectives and Pipeline

Learning Objectives. The learning objective of the AC-SUM-GAN architecture consists of some loss value of the training process of the aforementioned models. For instance, a regularization loss ($L_{sparity}$), based on a regularization factor σ to force the State Generator to create a spare and diverse set of frame-level scores. A prior loss (L_{prior}) indicates the amount of lost information when using the Encoder's latent space to represent the VAE's prior distribution. A reconstruction loss (L_{recon}) computes the distance between the two feature vectors, one is from the original video and the other is from the reconstructed video. A pair of "original" loss (L_{ORIG}) and "summary" loss (L_{SUM}) are used to train the Discriminator in a label-based approach by labeling "1" for the original video's feature vectors and "0" for the reconstructed video's feature vectors. The former is used to maximize the similarity between compressed feature vectors of the original video and the new latent representation of it. Similarly, the latter is used to maximize the similarity between the compressed feature vectors of the

[16]Kernel Temporal Segmentation

generated video and the latent representation of it. Finally, an L_{GEN} is a measurement of the difference between the latent representations, which is computed by the Discriminator when it is fed with the compressed feature vectors of both original and generated videos. Therefore, it indirectly forces the Generator to produce realistic video frames that look identically alike to the origin video.

As we have detailed introduced about the training process of the AC model, the feedback from the Critic is used as the output to train the Actor in order to figure out the policy that optimizes the probability in selecting key fragments to form the summary. The Actor's loss is seen as an evaluation value for this process and computed as follows:

$$L_{actor} = -\frac{1}{N}(\Sigma_{i=1}^{N} \ln c_i \alpha_i + \delta \Sigma_{i=1}^{N} H(c_i)) \quad (16)$$

where $\ln c_i$ and $H(c_i)$ represent the logarithm and the entropy of the computed density function c_i in each iteration i, the advantage α_i is a measurement of how well the Actor's choice at i^{th} step is compared to the average action at the same step. Finally, δ is an entropy regularization coefficient. The formula to compute the advantage α_i is represented as follows:

$$\alpha_i = z_i - v_i, \; i \in [1, N] \quad (17)$$

where v_i is the assessment value which is computed by the Critic, z_i is the discounted cumulative reward of all steps and is calculated as the sum of the multiplication between the Discriminator's reward at a step i of the game and the exponential of the discount factor γ ($\gamma \in R$, $0 \leq \gamma \leq 1$), which shows how important the future rewards means to a specific action choice of the Actor at the current state, from the current step to the N step:

$$z_i = \Sigma_{k=i}^{N} \gamma^{k-i} r_i \quad (18)$$

Finally, the process of training the Critic to learn how to evaluate the Actor's choice at iteration i^{th} by computing the v_i value is optimized by the following formula:

$$L_{critic} = \frac{1}{N} \Sigma_{i=1}^{N} \alpha_i^2 \quad (19)$$

Learning Pipeline. The training process of the AC-SUM-GAN architecture includes four steps and in each of the steps, it occurs in a different part of the architecture. Specifically, in the 1^{st} step, the algorithm makes a forward pass through the entire network to compute L_{prior} and L_{recon}. After that, it makes a backward pass update for the Encoder.

The 2^{nd} begins when the algorithm runs through the partially updated model to compute L_{recon} and L_{GEN}, then it uses the sum of those two to update the Decoder.

The 3^{rd} step consist of two sub-steps. The first one is a forward pass through the model once again and results in the feature vectors $\hat{X}$ of the reconstructed video, then those feature vectors are used to compute L_{SUM}. After that, the compressed feature vector of the original video X' are fed to the Discriminator to calculate L_{ORIG}. The second sub-step

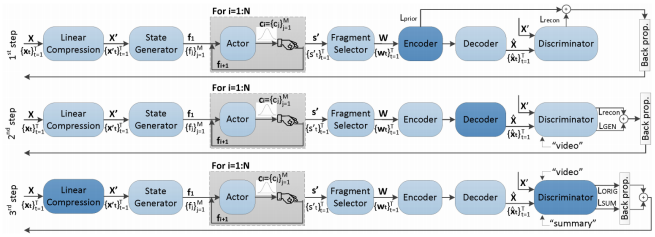


Fig. 3. The first three steps of the pipeline

is taking the derivation of the losses through two individual backward passes to receive the gradients and used them to update the Discriminator and the linear compression layer which subsequently affects the compressed feature vectors.

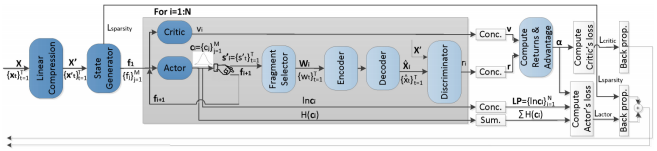


Fig. 4. The fourth step of the pipeline

The 4^{th} step, which is the final one shown in figure 4, carries out the training processes of the State Generator and the AC model. Specifically, the original feature vectors X are compressed by the linear compression, then it goes through the State Generator to initialize the starting state ($f_1 = \{f_j\}_{j=1}^{M}$) of the action-state space. After that, the AC model begins an N iterations loop of the selecting key fragments process. The Actor roles is to generate a distribution c_i and sample an action from it, while the Critic computes an evaluation value v_i to judge if the taken action was good or not. The taken action affects the frame-level scores s and produces new scores s'. Finally, the Discriminator will evaluate the generated video by giving it a reward r_i at each step of the game.

After the game ends, the architecture outputs the vectors $v = \{v_i\}_{i=1}^{N}$, reward $r = \{r_i\}_{i=1}^{N}$, $LP = \{\ln c_i\}_{i=1}^{N}$, and the scalar value $En = \Sigma_{i=1}^{N} H(c_i)$. The first two elements are used to calculate the discount cumulative reward of all steps and then the advantage. The computed advantage contributes in the training processes of the Actor and the Critic. The training processes of the Actor and the State Generator is simultaneously performed. The advantage $\alpha = \{\alpha_i\}_{i=1}^{N}$, LP and En values are used to compute the Actor's loss L_{actor} that later on is used to train the Actor, $L_{sparity}$ is used to train the State Generator. The linear compression also trained in this step.

The figure 5 indicates that the step-wise learning process applied to the architecture makes the training process for every component becomes more effective overtime, the longer the training process occurs the higher the rewards it gets. The horizontal axis indicates the number of epoch.

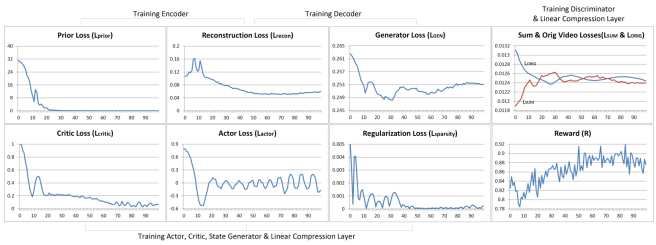


Fig. 5. Loss and reward curves for the proposed model

IV. Experiment

A. Dataset

Both our models are trained on the dataset TVSum [6] and SumMe [5]:

- **TVSum** is a dataset that comprises 50 general-purpose videos, which was artificially noted. 20 humans were asked to grade a concise 2-second video segments, with the score from 1 if it was dull to 5 if it was interesting. The frame-level scores is consequently computed by giving a score to each frame in the video segment. They are also rescaled in [0,1] using min-max normalization.

- **SumMe** was released as a publicly available, human-annotated Video Summarization benchmark dataset for frame importance score prediction. Inside this folder has a collection of 25 general-purpose public videos from YouTube, with each video has 15 to 18 human annotations. They were asked to make a summary having the foremost important content with a limited time length. Thus, the annotations are "0" or "1" values for every frame of the videos, indicating if a selected frame was a component of his summary. As a target for the models, importance frame scores are derived from the proportion of annotators who picked the frames, and most of the scores follow a volatile pattern between 0.0 and 0.5.

In addition to the above 2 datasets, the DSN model also exploits 2 other datasets, respectively, OVP1 with 50 videos and YouTube [8] with 39 videos excluding animated videos to evaluate the augmented training data in their model settings.

B. DSN Results

Evaluation metric and setting. For a fair comparison between the DSN and other approaches, the author follows the common protocol from [7] to generate an F-score as a measure of the similarity between automatic summary and ground-truth summary.

When it comes to the setting, includes three parts as suggested in [7] to evaluate:

1. **Canonical**: the standard 5FCV[17], i.e., the ratio of videos for training and testing corresponds to 4:1.
2. **Augmented**: 5FCV and the improved training data by OVP and YouTube for each fold.
3. **Transform**: With a target dataset (SumMe or TVSum), using the other three datasets as training data to test transferability.

Implementation Details. Instead of using the source code in the author's article, we replaced another source code that was modified based on the sample source for testing. However, the parameter settings remain unchanged. We also downsample videos by 2 fps and set the temporal distance to 20, still the length ratio ϵ is 0.5. The quantity of episodes N to 5, as well the other hyperparameters, with α, β_1 and β_2 are optimized via cross-validation. We set the dimension of hidden state within the RNN cell to 256 throughout this paper.

After completing the training and getting the final result, we will divide it into 2 main comparison contents. The first is to compare our own model with baselines for different learning objectives. Then we compare our methods with the most state-of-the-art unsupervised/supervised approaches.

Comparison with baseline. We first start to set the main baseline model trained only with a reward, R_{div} and R_{rep} respectively, with the former is denoted by D-DSN while the latter is R-DSN. After calculating each reward separately, we combine them into a reward function and let the baseline model trained with this function as it used to, which is named DR-DSN. The model also extends to the supervised version called DR-DSN$_{sup}$. Recall a lambda function above, the author validates the effectiveness of the λ-technique ignoring the distant similarity when computing R_{div}. We continue to take the D-DSN model trained without λ-technique, with the symbol as D-DSN$_{w/o\lambda}$. To demonstrate that DSN can get more benefit from reinforcement learning than from supervised learning, the DSN$_{sup}$ model is generated by training with cross entropy loss using key frame annotation, where a regularization term is interpreted as a confidence penalty for a distributed output.

Method	SumMe	TVSum
DSN$_{sup}$	32.3	55.5
D-DSN$_{w/o\lambda}$	33.1	56.1
D-DSN	33.4	56.9
R-DSN	33.7	57.5
DR-DSN	34.7	58.4
DR-DSN$_{sup}$	**35.7**	**59.1**

TABLE I
RESULTS (%) OF DIFFERENT VARIANTS OF OUR METHOD ON SUMME AND TVSUM

The table elucidates the results of various variants within the DSN model on both TVSum and SumMe. The initial impression from the table is that the DR-DSN unsupervised

[17]5-Folder Cross Validation

model performed significantly better than the DSN$_{sup}$ supervised model, with a rate of about 2.4% and 2.9% for the SumMe and TVSum datasets, respectively. It also justifies the belief in relation to DSN that reinforcement learning will help the model gain more benefits than supervised learning. Additionally, there's the same upward trend in the difference between DR-DSN and a pair of variants R-DSN and D-DSN, respectively, about 1.5%, showing the outer-performance of the model when collaborating with both two reward functions (R_{div} and R_{rep}) will produce high-quality, diverse, and representative summaries. Adding L_{MLE}, a supervised signalization to the DR-DSN model resulted in a marked improvement in summarization performance (with 35.7% on SumMe and 59.1% on TVSum).

In general, the performance of R-DSN is better than that of D-DSN because diversity summaries often contain redundant information unrelated to the original video. Not only that, but when D-DSN does not have a lambda, the model performance is not better than that of D-DSN because of no consideration in temporally distant frames. Choosing frames that are far away temporarily can lead to higher reward, so this is why DSN is encouraged to apply lambda technique with diverse reward function in training. About 50%-70% of the distance matrix is set 1 for the early stages, and this percent also increases in parallel with the training epochs, but usually keep around 80%-90%.

Comparison with the author's result. As can be seen from the tables, it is obvious from our chart that TVSum F-score shows a significant score in all type of DSN experiment, while the figure for SumMe shows an opposite trend. Comparing our data with the original result in both type (SumMe and TVSum) using DSN model to train, there is a big difference in the SumMe results and our SumMe data is all lower than the author SumMe data. In terms of TVSum results, our data is higher than author data in 6 different DSN methods, although there is no considerable differences in the result. Because the model train uses random function to choose random epoch, our results is still objective.

Method	SumMe	TVSum
DSN$_{sup}$	38.2	54.5
D-DSN$_{w/o\lambda}$	39.3	55.7
D-DSN	40.5	56.2
R-DSN	40.7	56.9
DR-DSN	41.4	57.6
DR-DSN$_{sup}$	**42.1**	**58.1**

TABLE II
THE AUTHOR'S RESULT

Comparison with unsupervised & supervised approaches It could be observed from the table III that the DR-DSN model only seems to perform better than other approaches on the TVSum dataset (58.4%) for our results, whereas opposite patterns are evident in the table SumMe, with the current state-of-the-art GAN$_{dpp}$ surpass a tiny fraction of 4%. It is manifest from the table IV that the

Method	SumMe	TVSum
Uniform sampling	29.3	15.5
K-medoids	33.4	28.8
GAN$_{dpp}$	**39.1**	51.7
DR-DSN	34.7	**58.4**

TABLE III
RESULTS (%) OF THE UNSUPERVISED APPROACHES ON SUMME AND TVSUM

LTSM-based models Bi-LTSM, DPP-LTSM and GAN$_{sup}$, respectively, have F-scores are outperformed by the DR-DSN$_{sup}$ as well as DR-DSN model ranging from 2.1% to 4.9% on TVSum. Not only that, but table IV is matched by a roughly similar trend as table III with GAN$_{sup}$ beating DR-DSN$_{sup}$ at SumMe.

Method	SumMe	TVSum
Bi-LTSM	37.6	54.2
DPP-LTSM	38.6	54.7
GAN$_{sup}$	**41.7**	56.3
DR-DSN$_{sup}$	35.7	**59.1**

TABLE IV
RESULTS (%) OF THE SUPERVISED APPROACHES ON SUMME AND TVSUM

However, if we use SumMe results of DR-DSN and DR-DSN$_sup$ in the author's paper, we can confirm that both variant approaches outperforms the other unsupervised approaches on both datasets by large margins. Additionally, although the reward functions are conceptually similar to GAN objectives, the DR-DSN model gives better results as the direct modeling for the diversity and representativeness of selected frames in the feature space that leads to the DSN finding more and better solutions.

C. AC-SUM-GAN Results

Evaluation metric and protocol. A key-fragment-based approach proposed in [7], which is widely used in most SoA, using a score named F-Score (as percentage) to indicate overlap between the machine-generated summary and the handcrafted ground-truth summary. The SumMe dataset's summaries can be directly evaluated by the protocol, while the TVSum need to be transformed into key-fragment-based summaries. Finally, when giving a video in dataset to the model, it will generate out a summary a compare that summary to all the available human-crafted summaries to compute a set of F-Scores. SumMe dataset will output the maximum value of F-Score in the set, while TVSum produces the average of them.

The articles proposed a split proportion of 80:20 for the training and testing data, respectively. Moreover, the experiments will be performed on five different splits of the dataset, which will be generated randomly, and reported the average performance.

Implementation Details. Similar to most SoA methods, the authors of this GAN model downsamples the videos to 2 fps. The number of non-overlapping, equals in length video fragments and determined as the smallest unit in the video are notated as M, in this experiment is 60 frames, $M = 60$. The duration of a single fragments is notated as d, and it is computed by taking the total frames of the video and divide it by M. The length of the synopsis must not exceed 15% of the video's duration. Deep representations of frames were obtained by taking the output of the pool5 layer of GoogLeNet trained on ImageNet. The original feature vectors at size 1024 must be compressed down to 512. The State Generator, Encoder, Decoder and Discriminator are models that comprise a 2-layer LSTM with 512 hidden units, while the State Generator's LSTM is a bi-directional one. The Actor has 4 fully connected layers, whereas the Critic has 5. The output of the Actor's last layer is seen as the input to a softmax layer to produce a categorical probability distribution. The Critic's last layer produces a scalar value between 0 and 1. The discount factor is set to 0.99 with the aim of making the future reward more meaningful. The entropy regularization coefficient δ is set to 0.1. Finally, the model is trained in a full-batch mode and is optimized by the Adam algorithm. The learning rate of the Discriminator is set at 10^{-5} while the rest is at 10^{-4}. The training process will terminate if it reaches the limited number of epochs, which it set to 100 in this experiment. Lastly, the well-trained model is selected based on the criterion of maximize the received rewards and simultaneously minimize of the Actor's loss.

Selecting the Trained Model. The evaluation of a well-trained model and in this case the AC-SUM-GAN architecture based on some following criteria:

- The maximization of final reward, which computed by taking the average of the received rewards r_i at every step of the "N-picks" game.
- The maximization of the final reward and the minimization of the Actor's loss.
- The minimization of the reconstructed loss L_{recon} that measures the similarity between the original video and the reconstructed video.
- The minimization of both the reconstruction L_{recon} loss and sparsity loss $L_{sparsity}$
- The maximization of the final reward and the minimization of the reconstruction loss L_{recon}

The regularization factor σ in the experiment is set as a discrete variable and varies in range $[0.1, 1]$ with equaling steps 0.1. Instead of choosing a specific value for σ, the best value for σ is chosen based on the aforementioned criteria for model selection. Those criteria indicate both the training epoch and the σ, thus it is responsible to select a well-trained model.

Criterion/ Dataset	Reward	Reward & Actor loss	Recon. loss	Recon. & Sparsity loss	Reward & Recon. loss
Article's TVSum	60.5	60.6	60.7	60.8	60
Experiment TVSum	84.8	46.8	67.6	33.8	76.2

TABLE V
COMPARISON BETWEEN THE ARTICLE'S PERFORMANCE AND THE EXPERIMENT ON TVSUM DATASET BASED ON DEFINED CRITERIA. VALUES REPRESENT F-SCORE (%).

Evaluation Results and Comparisons with DSN. The

Method	TVSum
DR-DSN	58.4
AC-SUM-GAN	58.4

TABLE VI
COMPARISON RESULTS (%) OF DIFFERENT DR-DSN UNSUPERVISED LEARNING AND AC-SUM-GAN ON TVSUM DATASET

Method	TVSum
DR-DSN$_{sup}$	**59.1**
AC-SUM-GAN	58.4

TABLE VII
COMPARISON RESULTS (%) OF DR-DSN SUPERVISED LEARNING AND AC-SUM-GAN ON TVSUM DATASET

tables VI and VII illustrate the average F-Scores of the variants of DSN (Supervised Learning) architecture with AC-SUM-GAN (Unsupervised Learning) architecture. It can clearly be seen that the AC-SUM-GAN architecture produces a lower average F-Score than the DR-DSN$_{sup}$ but equal to DR-DSN. However, the number of epoch in the training process of AC-SUM-GAN is 100, while in both DSN methods is 120, thus the bias is not too much important. The special aspect of AC-SUM-GAN is that it completely eliminates the human impact on the training process, which takes a lot of effort to training a model. Secondly, this is the first to combines the AC model with GAN architecture and applies it in summarizing videos.

Ablation Study. This ablation study is a part that evaluate the contributions of the major components in the whole AC-SUM-GAN architecture, such as the Generator (Variational Auto-Encoder), the Discriminator and the Actor-Critic model. We consider the AC-SUM-GAN architecture in three different versions of it:

- **AC-SUM-GAN without VAE.** This version excludes the VAE, and directly feed the compressed feature vectors to the Discriminator. This variant only contains

the 3^{rd} and 4^{th} steps from figures 3 and 4.

- **AC-SUM-GAN without Discriminator.** In this version, instead of using the Discriminator to compute the distance between the compressed original feature vectors and the reconstructed feature vectors, they are directly compared to each other with a matrix operator and then compute the reconstruction loss. This process only occurs in step 3^{rd} of the pipeline and visualized in figure 3.

- **AC-SUM-GAN without Actor-Critic.** This version does not include the Actor-Critic model and the State Generator's function F(s), instead, the State Generator and the linear compression is updated through the sum of $L_{sparsity}$ and L_{recon}, which is shown in step 4^{th} (Figure 4).

To bring this study into experiment, the article chooses a fixed regularization factor σ equals to 0.5, which is the median value of σ, and the best trained model to apply on the same dataset with the same five randomly generated split of data. The result in table VIII indicates that the Actor-Critic model and the Discriminator made a huge impact in improving the architecture, whereas the VAE contributed a positive improvement.

	SumMe	TVSum
AC-SUM-GAN w/o VAE	53.0	61.1
AC-SUM-GAN w/o Discriminator	53.3	60.7
AC-SUM-GAN w/o Actor-Critic	50.4	60.7
AC-SUM-GAN	**54.5**	**61.4**

TABLE VIII
ABLATION STUDY BASED ON THE PERFORMANCE (F-SCORE (%)) OF THREE VARIATIONS OF THE PROPOSED MODEL, ON SUMME AND TVSUM.

Moreover, in order to estimate the computational complexity of the AC models, which embedded in the architecture, the article measures consumed time during the training and the latter processes. The result of the test shows that the whole architecture takes 55% more time than the version without the AC model. This extra time is used to learn the additional parameters, whereas the inference processes the same and the video summarization takes less than 0.2 seconds.

V. Conclusions

In this paper, we recommended an unlabeled reinforcement learning algorithm to solve the unsupervised video summarization problem. After performing many experiments on two popular benchmark data sets (SumMe and TVSum), illustrates the combination of unsupervised reward function and reinforcement learning outputs results equivalent or even superior to other modern unsupervised substitutes as well as supervised methods. Besides that, we also introduced an architecture that comprises a Generative Adversarial Network (GAN) and an Actor-Critic model that is first used in summarizing video. This method concentrates on improving the impact of the fragments selecting process on creating a brief synopsis of a video. The Actor is trained by receiving feedbacks from the Discriminator and the Critic. The learning pipeline allows the Actor to figure out the optimal policy for selecting key-fragments and the Critic to compute the value function. The paper also brings out some criteria to evaluate the model's performance based on two benchmarking datasets, SumMe and TVSum. The paper also comparing the architecture's result with the original result from the articles and the DSN methods. Finally, the ablation study reaffirms the benefit of embedding the AC model into the GAN architecture, which surely makes the model better.

VI. Acknowledgement

We would like to express our special thanks of gratitude to our teacher Dr. Ngoc Hoang Luong who gave us the golden opportunity to do this wonderful project on the topic Comparison between Reward Function with Deep Summarization Network and Actor-Critic with Generative Adversarial Networks for Unsupervised Video Summarization. We also want to send our appreciation to Mr.Sylvain Marchienne who has modified the DSN code, to Elena Adamantidou, Evlampios Apostolidis, and Alexandros Metsai who has provided us the source code of AC-SUM-GAN architecture, to Duong Huynh Huy, and Lam Thanh Tin who supported us in setting code.

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

[2] A. Goyal, N. R. Ke, A. Lamb, R. D. Hjelm, C. J. Pal, J. Pineau, and Y. Bengio, "ACtuAL: Actor-Critic Under Adversarial Learning," ArXiv, vol. abs/1711.04755, 2017.

[3] Kaiyang Zhou, Yu Qiao, Tao Xiang, Deep Reinforcement Learning for Unsupervised Video Summarization with Diversity-Representativeness Reward.

[4] Optimization Algorithms: Adam, https://d2l.ai/chapter_optimization/adam.html

[5] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool, "Creating Summaries from User Videos," in European Conf. on Computer Vision (ECCV) 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 505–520.

[6] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes, "TVSum: Summarizing web videos using titles," in 2015 IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR), June 2015, pp. 5179–5187.

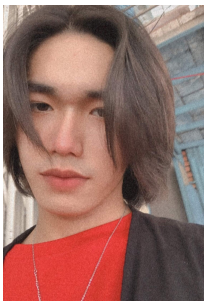
[7] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, "Video Summarization with Long Short-Term Memory," in European Conf. on Computer Vision (ECCV) 2016, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 766–782.

[8] De Avila, S. E. F.; Lopes, A. P. B.; da Luz, A.; and de Albuquerque Araujo, A. 2011. Vsumm: ´ A mechanism designed to produce static video summaries and a novel evaluation method. Pattern Recognition Letters 32(1):56–68.

VII. Biography

Truong Minh Chau graduated from Bui Thi Xuan high school in 2019. Currently, she is a sophomore majoring in Computer Science at University of Information Technology. She has co-authored 4 reports in fields of math for Computer Science, design and analysis of algorithms, video summarization, and machine translation.

Pham Minh Khoi graduated from Bui Thi Xuan high school in 2019. Currently, he is a sophomore majoring in Computer Science at University of Information Technology. He has co-authored 4 reports in fields of math for Computer Science, design and analysis of algorithms, traffic signs detection and classification, and video summarization.



Le Doan Thien Nhan graduated from Thu Khoa Nghia high school in 2019. Currently, he is a sophomore majoring in Computer Science at University of Information Technology. He has co-authored 4 reports in fields of math for Computer Science, design and analysis of algorithms, video summarization, and machine translation.