Attentive and Adversarial Learning for Video Summarization

Tsu-Jui Fu
National Tsing Hua University
s103062110@m103.nthu.edu.tw

Shao-Heng Tai
Umbo Computer Vision
daniel.tai@umbocv.com

Hwann-Tzong Chen
National Tsing Hua University
htchen@cs.nthu.edu.tw

Abstract

This paper aims to address the video summarization problem via attention-aware and adversarial training. We formulate the problem as a sequence-to-sequence task, where the input sequence is an original video and the output sequence is its summarization. We propose a GAN-based training framework, which combines the merits of unsupervised and supervised video summarization approaches. The generator is an attention-aware *Ptr-Net that generates the* cutting points *of summarization fragments. The discriminator is a 3D CNN classifier to* judge *whether a fragment is from a ground-truth or a generated summarization. The experiments show that our method achieves state-of-the-art results on SumMe, TVSum, YouTube, and LoL datasets with* ***1.5%*** *to* ***5.6%*** *improvements. Our Ptr-Net generator can overcome the unbalanced training-test length in the seq2seq problem, and our discriminator is effective in leveraging unpaired summarizations to achieve better performance.*

1. Introduction

The task of video summarization is to produce a shortened video clip that conveys the gist of its original longer version. Applications of video summarization include video highlight for sports games, movie recaps, and rare event detection in video logs for efficient browsing, *etc*.

Image features of various characteristics at different levels have been used in video summarization, for example, low-level visual consistency [4, 23] and high-level semantic changes [16, 36]. The video summarization method of [25] uses a pre-trained CNN model to extract image features and then performs clustering for keyframe selection.

On the other hand, recurrent neural networks (RNNs), in particular, long short-term memory (LSTM) units [13] are widely used to model and memorize sequential data. A video can be considered as a long image sequence, and LSTM has shown to be capable of deciding whether the current frame is a keyframe given the information of previous frames [8, 40].

AVS [15] view video summarization as a sequence-to-sequence (seq2seq) problem, in which the input sequence is a video and the output sequence is its summarization. They propose to apply an attention mechanism that assigns different importance weights to different input frames for more accurate summarization. Although RNN-based and seq2seq architectures are good at modeling long sequences, they often suffer from the problem of inconsistency between the lengths of the training and the test data [31], which makes the trained model hard to generalize to test data.

Another issue of applying supervised learning to video summarization is that it is hard to collect original-versus-summarized pairs. However, we can easily collect many summarization-like videos such as movie trailers or sports game highlights from Youtube and Twitch, and we may use them as guiding examples of summarization fragments even though we do not have the original videos.

Contributions: We present an adversarial training framework for semi-supervised video summarization. The framework employs an attention-based pointer network (Ptr-Net) [31] as the generator to predict the cutting (starting and ending) points for each summarization fragment. The discriminator in our framework is a 3D CNN classifier to judge whether a video fragment is a good summarization or not. An overview of our method is illustrated in Fig. 1. We train this framework as a generative adversarial network (GAN) [10], and show that our method achieves the state-of-the-art performance on many challenging benchmark datasets [2, 11, 28]. The main advantages and contributions of our method are twofold:

1. We build a Ptr-Net as the generator and show that it can effectively handle videos of widely varying lengths and thus resolve the issue of inconsistent lengths of training and test data. To our best knowledge, this is the first work that uses the Ptr-Net to address the length-inconsistency issue in video summarization.

2. Our method does not need more pairs of original and summarized videos for improved performance—the discriminator can strengthen the generator using unpaired summarization fragments available on the Web.

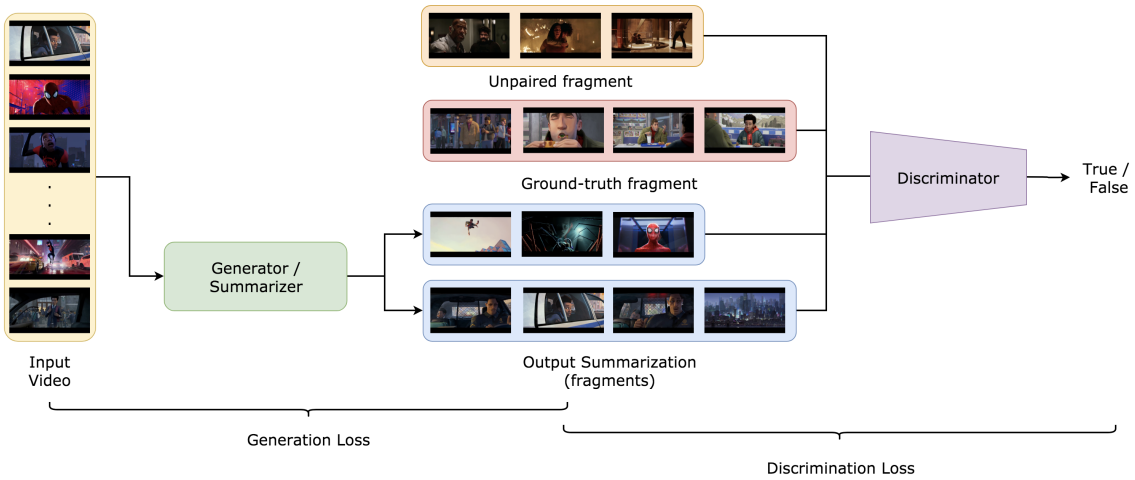

Figure 1. An overview of our method. We present a GAN-based approach to video summarization. The generator, which is implemented as a Ptr-Net, summarizes an input video into a summarization with video fragments. The Ptr-Net generator can handle the issue of inconsistency between the lengths of training and test data. The discriminator judges whether a summarization fragment is from the real summarization data or is produced by the generator. With the discriminator we may use other summarization-like videos as guiding examples even if their original videos are not available.

2. Related Work

From the learning perspective, we may roughly divide video-summarization methods into unsupervised approaches [1, 2, 5, 6, 7, 18, 20, 22] and supervised (or semi-supervised) approaches [8, 9, 15, 27, 37, 39, 41]. We give a quick overview below with respect to this dichotomy. Notice that our method belongs to the supervised/semi-supervised category.

A way to achieve unsupervised video summarization is to use domain-specific knowledge as the criterion for keyframe selection, for example, sports highlights with keyword "home run" [7]. The shortcoming of using domain-specific knowledge is that we need to handcraft different key features for each domain. Another way for unsupervised video summarization does not need domain-specific knowledge but relies on visual relevance. Criteria of visual relevance include content frequency [2, 18], coverage [5, 22], and user attention [6, 20].

Mahasseni *et al*. [21] propose a GAN-based method that contains a variational autoencoder (VAE) as summarizer to generate summarization and an RNN-based classifier as discriminator to distinguish original (unsummarized) or summarized videos. Their method achieves the state-of-the-art performance under the category of unsupervised video summarization. Note that, our discriminator is posed with a different task: it aims to judge whether a summarized video is from the summarization dataset or is generated by the summarizer.

Supervised video summarization approaches use the training pairs that consist of original videos and corresponding human-generated summarizations to learn how to summarize videos. For instance, Gong *et al*. [9] formulate the video summarization task as a supervised subset selection problem, proposing a sequential Determinantal Point Process (seqDPP) to learn how a subset is selected. Potapov *et al*. [27] train a support vector machine to assign each fragment a score and compose summarization from those of high scores.

A video can be considered as a sequence of image frames and thus is reasonable to be modeled by RNNs. Zhang *et al*. [40] use bidirectional LSTM to model dependency of image sequences and decide if a frame should be included as part of the summarization. Fu *et al*. [8] also present a similar method for the video-highlight task on League of Legend (LoL) highlight dataset. A critical issue of all the above methods is that they view the whole input frames as equally important and ignore the underlying temporal structures.

To solve this problem, AVS [15] and re-SEQ2SEQ [41] formulate video summarization as a seq2seq problem in which the input sequence is an original video and the output sequence is a summarization. Further, they incorporate an attention mechanism into the seq2seq architecture, assigning difference importance weights to different frames and achieving a better performance. Frames are selected into the summarization set according to their importance weights, and therefore, during training, the method of AVS needs each frame's score as the ground-truth importance weight. However, in practice, it is very difficult to collect video data along with the per-frame scores for such a formulation of training.

Since original videos and summarized sequences are usually very long, the unrolling in seq2seq decoder would thus be also very long, which makes the network hard to

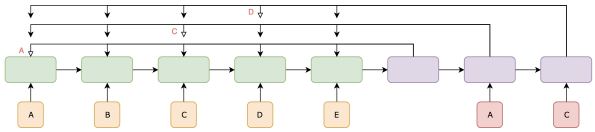

Figure 2. An overview of Ptr-Net. The output of each decoding time is the softmax distribution over all inputs, and the pointed inputs (*e.g.*, 'A' and 'C') will also be used for the next decoding.

train and degrades the performance for long summarization. Another drawback of previous seq2seq decoders is the inconsistent lengths of training and test videos, which results in worse performance during testing. Our method avoids the issue of varying lengths by building a Ptr-Net [31] as the seq2seq generative model. Instead of extracting fragments of consecutive frames from the input video, we only predict the cutting points, *i.e.*, a tuple of starting and ending points for each summarization fragment, so that the output unrolling could be efficiently encoded. Moreover, we combine the merits of supervised and unsupervised approaches by designing a discriminator that can train with either original-summarized pairs or merely unpaired summarization fragments to improve the summarizer.

3. Review of Ptr-Net

Ptr-Net [31] is one kind of seq2seq architecture whose decoder selects a member from the input sequence as the output, as show in Fig. 2. A typical seq2seq model [29] consists of two RNNs: an encoder and a decoder. The goal of the seq2seq model is to estimate the conditional probability of output sequence given the input: $P(y_1, ..., y_{T'}|x_1, ..., x_T)$, where $(x_1, ..., x_T)$ is the input sequence, $(y_1, ..., y_{T'})$ is the output sequence and the length T' of the output may be different from the length T of the input. The encoder, which is an RNN and usually uses LSTM cells for better memorization, first encodes the input sequence $(x_1, ..., x_T)$ into a fixed-dimension representation h_T given by the last hidden state of the RNN. Then, the decoder computes the probability of $(y_1, ..., y_{T'})$ with the initial hidden state set to h_T:

$$P(y_1, ..., y_{T'}|x_1, ..., x_T) = \prod_{i=1}^{T'} P(y_i|h_T, y_1, ..., y_{i-1}) \,. \quad (1)$$

Each $P(y_i|h_T, y_1, ..., y_{i-1})$ distribution function is implemented as a softmax over all outputs. The attention model [3, 35] augments the the encoder and decoder using attention mechanism over the entire sequence of encoder. Let the encoder hidden states be $(h_1, ..., h_T)$ and the decoder hidden states be $(d_1, ..., d_{T'})$, and we compute the attention vector at each output time i:

$$m_j^i = v^T \tanh(W_1 h_j + W_2 d_i) \,, \quad a_j^i = \text{softmax}(m_j^i) \,, \quad (2)$$

where m_j^i is the attention weight for input j at output time i, and a_j^i, derived from m_j^i after softmax normalization, is the attention mask over input j. Further, v, W_1, and W_2 are learnable parameters for the attention mechanism.

Ptr-Net is a special attention-based seq2seq model where the output dictionary size depends on the number of elements in the input sequence. Similar to seq2seq with attention mechanism, Ptr-Net explicitly uses the attention mechanism to obtain the the output distribution, a^i, represented as a vector over all inputs j:

$$P(y_i|h_T, y_1, ..., y_{i-1}) = a^i \,. \quad (3)$$

Then, we can use greedy sampling or other sampling methods to choose the position in input as our final output prediction based on a^i.

Ptr-Net specifically targets on problems whose outputs are discrete and correspond to positions in the input, and therefore is very suitable for video summarization. Moreover, since Ptr-Net's output softmax is not restricted to a fixed size, it is able to work well with input sequences of varying lengths [31].

4. Main Idea

The proposed method for video summarization is a GAN model [10] that contains a Ptr-Net [31] as the generator and a 3D CNN binary classifier as the discriminator, as shown in Fig. 1. The generator summarizes the input video into several fragments, and the discriminator distinguishes whether a fragment is from ground-truth summarization or is generated by the summarizer. The goal is to make the generated summarization fragments as authentic to the discriminator as possible.

4.1. Ptr-Net Generator

We formulate video summarization as a seq2seq problem. The input sequence is an original video and the output sequence comprises the summarization fragments. Inspired by the question answering task in [32], we create a Ptr-Net generator that uses bi-directional LSTM [24] as the encoder and an attention-based cutting point predictor as the decoder. The decoder does not produce consecutive frames as the output but only tuples of the starting and ending points of fragments.

This way, we only need to maintain very compact information about the fragments. The Ptr-Net generator would not suffer from the difficulty of producing longer output sequences and would be easier to train. Our Ptr-Net generator is illustrated in Fig. 3.

We first use a pre-trained CNN model to extract a feature vector f_j for each input frame j. The encoder takes input sequence $\{f_j\}$ and encodes feature vector f_j into hidden states h_j and h'_j, using bi-LSTM RNN. The decoder

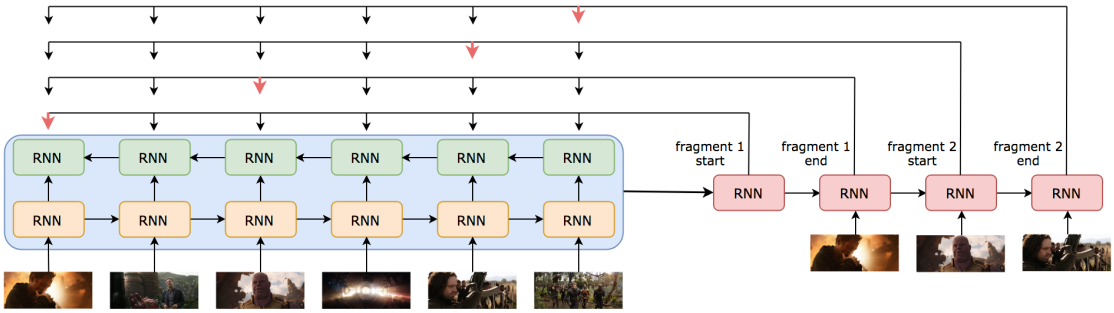

Figure 3. The Ptr-Net summarization generator: We use bi-LSTM as the seq2seq encoder. The point decoder performs as a frame selector (video cutter), and it only outputs the tuple of starting and ending points for each fragment, making the output sequence more compact and easier to train.

computes an attention mask m_j^i based on all input hidden states $\{h_j\}$, $\{h'_j\}$ and the decoding hidden state d_i at each decoding time i. Finally, the decoder produces a position distribution a^i (represented as a vector) for decoding time i over the input sequence $\{f_j\}$ according to the attention mask after softmax normalization.

$$f_j = \text{CNN}(\text{frame } j), \quad (h_j, h'_j) = \text{RNN}(f_j),$$
$$m_j^i = v^T \tanh(W_1(h_j, h'_j) + W_2 d_i),$$
$$P(o_i|h, h', o_1, ..., o_{i-1}) = a^i, \quad a^i = [\text{softmax}(m_j^i)],$$

where $P(o_i|h, h', o_1, ..., o_{i-1})$ is the predicted position distribution for each decoding time i, and W_1, W_2, and v are learnable parameters for attention based point decoder. We use the vector a^i to denote the aggregated activations of the softmax normalization m_j^i. We can view a^i as an output distribution over the position j in the input sequence.

Since we have the original video and its corresponding ground-truth summarization pair, we can train our generator via supervised learning. The output of the generator is represented as tuples. The decoder predicts in turn the starting point of fragment 1, the ending point of fragment 1, the starting point of fragment 2, the ending point of fragment 2, ..., and so on, with a finishing token denoted as '#' at the end. Similar to the typical seq2seq, we use a maximum output length L and apply paddings for all output sequence. Take, for example, a ground-truth summarization that contains tuples for fragments (1, 6) and (9, 11) and the maximum output length $L = 8$, the output sequence should be represented as follows:

$$\text{output: } \underline{1} \quad \overline{6} \quad \underline{9} \quad \overline{11} \quad \# \quad \# \quad \# \quad \#. \tag{4}$$

In our implementation, we add a blank image at the end of all frames in input video sequence, and if the decoder points to it, it will generate the finished token '#'. Since the length of generated summarization could be very long with respect to long input video sequence, we only output tuples of the starting and ending points for selected fragments instead of all frame indexes. This mechanism can prevent L from being too large and helps to avoid too many redundant finishing tokens padded to the tail of the output sequence.

The decoder produces a softmax distribution over input positions, and we can calculate loss via a categorical loss function with ground-truth positions represented as one-hot vectors. During training, we use *teacher forcing* [34] to train our point decoder. That is, no matter what we predict before, we always take the ground-truth position frame from the input for each decoding time. In training infancy, the generator predicts not so well, and if we do not apply teacher forcing, the previous prediction error would accumulate along the prediction, resulting in unstable training behavior on the whole sequence. Further, [34] proves that teacher forcing can help a sequence-generation model to converge faster. Another benefit of teacher forcing is that since the decoding input is restricted by the ground truth, we do not need to deal with the out-of-order problem such as the fragment (9, 11) coming out first and then the fragment (1, 6). Please note that we only apply teacher forcing at the training time; during inference, we simply do standard RNN unrolling.

In brief, our generator aims at minimizing the average of output categorical loss (J_1) for each time step:

$$J_1 = \frac{1}{L}\sum_{i=1}^{L}[\text{CategoricalLoss}(P(o_i), g_i)]\,, \tag{5}$$

where L is the length of output sequence, $P(o_i)$ and g_i are the predicted position distribution and the one-hot vector of ground-truth position at time i, respectively. In our experiments we use cross entropy as the categorical loss function.

4.2. Discriminator

A typical GAN model contains a discriminator to distinguish the ground-truth samples from the generated ones. The discriminator is trained with the generator under an adversarial learning mechanism. In our method, as shown in Fig. 4, the discriminator is a 3D CNN binary classifier [14]

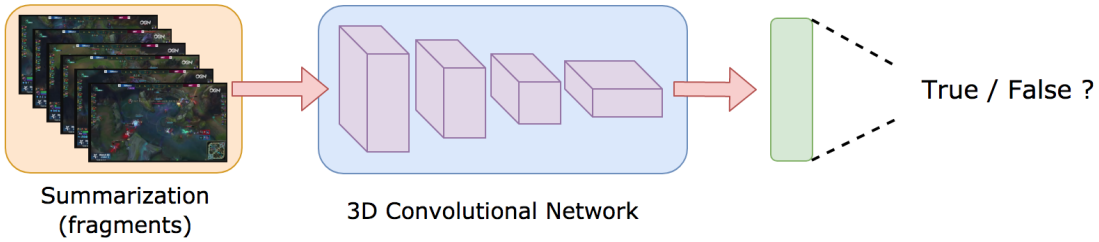

Figure 4. The summarization fragment discriminator. We use 3D CNN as a binary classifier to decide whether a fragment is from the ground-truth dataset or is generated by our summarizer.

that judges whether a summarization fragment is from the ground-truth dataset or is generated by our summarizer.

Instead of using RNNs again, we use a CNN architecture to construct the discriminators. CNNs are good at extracting dominant receptive fields, and we thus consider that it should be sufficient for a CNN to tell the authenticity simply based on several frames of a fragment. We randomly pick a fixed number of frames chronologically in a fragment, and concatenate the frames into a 3D cuboid as the discriminator input. Then, the cuboid goes through the 3D CNN and the output is a prediction on the cuboid being from the ground-truth summarization (True) or from the generator (False). The discriminator targets on maximizing the prediction correctness (J_2):

$$J_2 = \mathbb{E}_{x\sim \text{gt}}[\log(D(x))] + \mathbb{E}_{\hat{x}\sim \text{summarizer}}[\log(1 - D(\hat{x}))], \quad (6)$$

where x is a ground-truth summarization fragment and $\hat{x}$ is a summarizer-generated fragment.

4.2.1 Training the Ptr-Generator via Policy Gradient

Since the outputs of the ptr-generator are discrete probability distribution and are not differentiable, we cannot update our generator from the discriminator via simple back-propagation. Here, like SeqGAN [38], we adopt *policy gradient* to estimate the approximate gradient. That is, we consider the output from the discriminator as the reward and use REINFORCE algorithm [33] to train our generator so that it can maximize the reward from our discriminator.

The objective function (J_3) we want to maximize is the expected reward as follows:

$$J_3 = \mathbb{E}_{P(o_{1:L})}[R], \quad (7)$$

where P means the probability, o_i means the output position at decoding time i, $P(o_{1:L})$ represents $\Pi_i P(o_i|o_{1:i-1})$ and we set R as the true label output from the discriminator. We then estimate the gradient by running M samplings with the REINFORCE algorithm:

$$\begin{aligned}\nabla J_3 &= \sum_{i=1}^{L} \mathbb{E}_{P(o_{1:L})}[\nabla \log P(o_i|o_{1:i-1})R] \\ &\approx \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{L}[\nabla \log P(o_i^m|o_{1:i-1}^m)R^m],\end{aligned} \quad (8)$$

where the superscript m denotes that it belongs to the m-th sampling.

Since the approximation of ∇J_3 is unbiased, it may have very high variance [33]. One common way to reduce this variance is to subtract a baseline value b from the reward function R, transforming the approximated gradient into

$$\nabla J_3 \approx \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{L}[\nabla \log P(o_i^m|o_{1:i-1}^m)R^m - b^m]. \quad (9)$$

Here we apply the same baseline strategy as Lewis *et al.* [19], treating the bias value b as the average reward from then until now.

Although the discriminator can judge whether a fragment is suitable or not, doing summarization using only the discriminator is usually not good enough. Take a singing performance video for example. Each frame contains the singer can be expected as good material for summarization. Nevertheless, we would not pick all of them since we might try to avoid too many repeated or similar views in the summarization. Moreover, whether a fragment belongs to the summarization or not is not only based on its own content but also related to the entire original video, and hence we cannot predict accurately if we only judge the fragments one by one.

In general, the Ptr-Net generator learns how to select fragments from original videos, and the discriminator provides a common judgment about how a good fragment should be. Both of them are indispensable and in Section 5, we show that the performance can be improved after training with the discriminator.

4.2.2 Learning from Unpaired Video Summarizations

We can easily collect lots of summarization videos from YouTube and Twitch. Although the original videos are not

available to form training pairs, we may use only the summarization videos to train the discriminator as well as the generator under our GAN framework. For those unpaired summarization videos, since we do not know the exact split position for each fragment, we apply naive frame clustering like [25] to split whole video summarization into several fragments. We can use these unpaired summarization fragments as the ground-truth samples to train our discriminator by maximizing J_4:

$$J_4 = \mathbb{E}_{x' \sim \text{unpaired}}[\log(D(x'))] \,. \tag{10}$$

4.3. Optimization

Finally we can optimize our ptr-generator and discriminator by minimizing the following objectives:

$$\begin{aligned} &\text{Generator: } J_1 - J_3, \\ &\text{Discriminator: } -J_2 - \alpha J_4, \end{aligned} \tag{11}$$

where J_1 is the supervised categorical loss for the ptr-generator, J_3 is the expected reward from the discriminator via policy gradient, J_2 and J_4 represent the optimized targets for ground-truth and unpaired summarization fragments, respectively, and α is a weighting factor between ground-truth and unpaired for the discriminator. Please note that J_3, J_2, and J_4 are to be maximized, so we add negative signs in the objectives for minimization.

4.4. Inference

During inference, we only perform the generation part, which includes visual feature extraction and ptr-generation, to process the input video. Furthermore, when performing inference, we do not adopt *teacher forcing* as we have done during training, and therefore the output of predicted summarization fragments may be overlapped or out of ordering. Here we apply an intuitive post-processing step in which we choose the union of predicted fragments and sort them to meet the time order. For example, if the predicted outputs are (12, 16), (3, 7), and (6, 10). We will merge (3, 7) and (6, 10) into (3, 10) and then sort them to obtain (3, 10) and (12, 16) as the final summarization fragments.

5. Experimental Results

This section begins with a description of experiment setting. Then, we show the benchmark results with detailed analysis. Finally, except for the quantitative results, we also demonstrate our qualitative results.

5.1. Experiment Setting

5.1.1 Datasets

We evaluate our method on four datasets: SumMe [11], TVSum [28], YouTube [2], and LoL [8]. SumMe [11] consists of 25 recording event videos such as festival and sports, and TVSum [28] includes 50 YouTube videos. Both SumMe and TVSum provide frame-level importance score for each video. To compare with previous state-of-the-art methods, we follow [40] to convert frame-level score to keyframe summaries for evaluation. YouTube [2] contains 50 videos from Open Video Project (OVP), including animation, news, and sports. Since YouTube only provides selected keyframes as ground truths, we set them as our evaluation target directly. LoL [8] provides 218 videos about match highlight of League of Legends from NALCS. Instead of providing only keyframes, LoL provides the complete summarization videos. Table 1 shows the properties of these datasets.

Table 1. Datasets for evaluation.

Dataset	# Videos	Duration	Summarization Type
SumMe [11]	25	2-7	Keyframe selection
TVSum [28]	25	1-5	Keyframe selection
YouTube [2]	50	1-10	Keyframe selection
LoL [8]	218	30-50	Fragment selection

5.1.2 Implementation Details

For fair comparison with the previous methods, we use the output of pool5 layer of a pre-trained GoogleLeNet [30] (dimension=1024) as visual feature for SumMe, TVSum, and YoutTube. For LoL, we choose ResNet-34 [12] (dimension=512), the same as [8], to be our visual feature extractor.

Since we have two summarization types, keyframe selection and fragment selection, for different datasets, we implement different settings for each type. For keyframe selection on SumMe, TVSum, and Youtube, our point decoder of generator predicts the keyframe position directly and we see the whole keyframe set as summarization fragment to feed into the dicsriminator. For fragment selection on LoL, we maintain our setting as described in Section 4. We use 256 hidden units for both the encoder and decoder LSTM cells.

We implement a cross-entropy loss as the categorical loss function for optimizing the generator. For the discriminator, we construct five 3D convolutional layers and three fully connected layers as a 3D CNN classifier, and set the weighting factor α in Eq. (11) to be 0.6. For the policy gradient to train the ptr-generator, we run 16 samples each time, where is the M in Eq. (8). We train our proposed model with batch-size 16 on a GTX1080 Ti GPU and adapt Adam [17] optimizer whose learning rates for generator and discriminator are 0.005 and 0.002, respectively. Our model is implemented in PyTorch [26] and the code will be released after the review process.

Table 2. Performance (F-score) comparison with state-of-the-art methods on various datasets.

Method	SumMe	TVSum	YouTube	LoL
vsLSTM [40]	37.6	54.2	-	-
dppLSTM [40]	38.6	54.7	-	-
Li *et al*. [37]	43.1	52.7	-	-
Zhang *et al*. [39]	40.9	-	61.0	-
SUM-GAN [21]	41.7	56.3	62.5	-
A-AVS [15]	43.9	59.4	65.8	-
M-AVS [15]	44.4	61.0	66.2	-
Fu *et al*. [8]	-	-	-	72.2
Ours$_G$	45.5	60.8	68.2	75.1
Ours	**46.2**	**63.6**	**69.7**	**77.8**

5.1.3 Evaluation Metrics

We consider F-score, which is widely used in video summarization tasks, for evaluation [8, 15, 21, 37, 39, 40], that is,

$$P = \frac{S_{\rm gt} \cap S_{\rm pd}}{|S_{\rm pd}|}, \quad R = \frac{S_{\rm gt} \cap S_{\rm pd}}{|S_{\rm gt}|}, \quad F = \frac{2PR}{P+R}, \quad (12)$$

where $S_{\rm gt}$ and $S_{\rm pd}$ are ground-truth summarization and generated summarization, respectively. The final F-score is computed as F.

5.1.4 Baselines

We compare our method with several state-of-the-art video summarization methods, including both unsupervised and supervised approaches, and their results are all from the original papers. We choose supervised version of SUM-GAN [21] which is a GAN-based summarization method, vsLSTM [40] which uses RNN to model a video as a long image sequence, and AVS [15] which applies attention mechanism to frame importance weighting. We also include additional supervised methods such as [37], which learns the keyframe property weights, and [39], which transfers the summary structure from training videos to test videos. For fragment-selection video summarization task on LoL dataset, we compare our method with [8], which also uses RNN to model video frame sequences.

5.2. Quantitative Results

Table 2 shows the comparison results. Our method achieves the best performance. We train two variants of our method, one only with the Ptr-Net generator (Ours$_G$) and the other with the complete generator-discriminator model (Ours). We can see that just using the attention-aware Ptr-Net as summarizer (Ours$_G$) already almost improves the summarization results on all datasets, because Ptr-Net can give a more concise output in the form of the cutting point in the input sequence. The complete model with the discriminator (Ours) can further improve the performance, showing that our discriminator gives proper feedback to the generator and helps it to predict better.

For frame selection, our discriminator gets a 2.8% improvement on TVSum, better than on SumMe and YouTube. A possile reason is that since TVSum is collected from 10 different categories, it is not easy to learn the attention mask on so diverse inputs given only 25 videos. However, with the discriminator, the 3D CNN classifier can still classify the keyframe set very well, sending the additional feedback to train the generator better.

For fragment selection, we can also see a significant improvement over the previous method [8]. The main problem of [8] is that they select a frame by modeling only the previous frames, but a summarization usually also relates to the future consequence. For example, in LoL, a highlight usually happens during fighting. However, the method of [8] can only cut the fragment starting on fighting, but the ground-truth summarization usually includes several previous frames before the fighting really starts. Another example is common in sports videos like basketball. Someone making a jump shot can be a highlight, but the highlight should also include the player dribbling before jumping. Our Ptr-Net seq2seq generator sees the whole sequence and then predicts the starting and ending positions, and so we do not encounter that problem.

Table 3. Performance (F-score) comparison under unbalanced training-test length.

Dataset Train/Test	TVSum < 3.5 / > 3.5	LoL < 45 / > 45
vsLSTM [40]	51.6 (-2.6)	-
dppLSTM [40]	51.7 (-3.0)	-
Fu *et al*. [8]	-	70.7 (-1.5)
Ours$_G$	60.2 (**-0.6**)	75.0 (**-0.1**)
Ours	62.9 (**-0.7**)	77.3 (**-0.5**)

5.3. Detailed Analysis

5.3.1 Unbalanced Train-test Lengths

Typical RNNs suffer from a problem called unbalanced train-test length [31], especially when the length of input during testing is much longer. We do an experiment of unbalanced train-test length on TVSum and LoL datasets. For TVSum, we select videos shorter than 3.5 minutes as the training set, and test on the videos longer than 3.5 minutes. For LoL, we choose 45 minutes as the separation threshold.

The results are shown in Table 3. Both [40] and [8] have a performance drop under unbalanced train-test length. Since our Ptr-Net generator has a flexible attention point mechanism, it is not restricted to the input length. Our method achieves similar performance even with unbalanced train-test length.

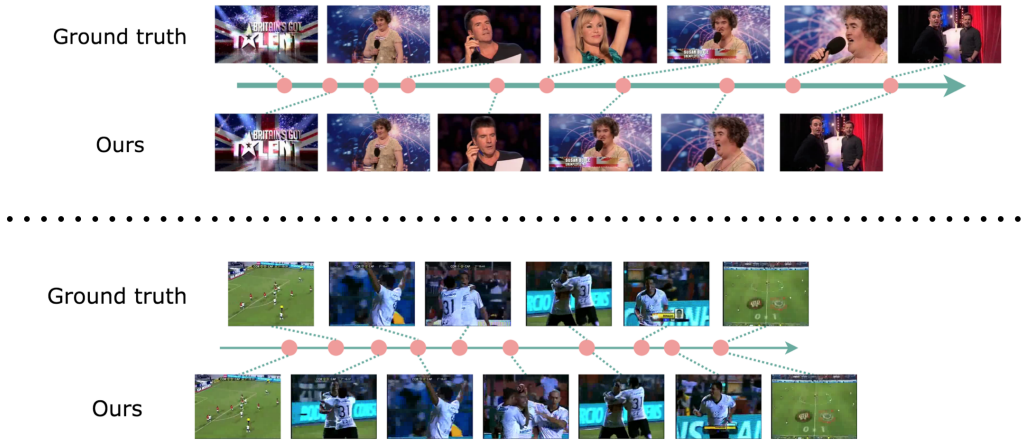

Figure 5. Examples of video summarization of YouTube dataset, along with the ground-truth summarization.

Table 4. Performance (F-score) improvement when trained with unpaired summarizations.

Dataset	Ours
SumMe + YouTube$_S$	47.3 (**+1.1**)
TVSum + Youtube$_S$	65.1 (**+1.5**)
LoL (NALCS) + LoL$_S$ (LMS)	79.4 (**+1.6**)

Table 5. Timing results of our proposed method under different datasets and batch-size.

Dataset	Batch-size	Duration (min)	Time (sec)
Youtube	1	3.95	11.69
Youtube	16	3.56	145.83
LoL	1	48.52	172.57
LoL	16	42.74	2218.15

5.3.2 Training with Unpaired Summarizations

Our model can also be trained with unpaired summarizations that do not have the original videos. We add only the summarization part of YouTube dataset, marked as YouTube$_S$, into SumMe and TVSum. Likewise, we add the summarization-only LMS part of LoL, marked as LoL$_S$ (LMS), into LoL dataset. As shown in Table 4, all of them gain improvement under the above settings. Please note that the summarization part is only for additional training, we still test on the same testing set as prior experiment.

5.3.3 Time Complexity

Table 5 shows the average length of video input and the overall processing time over a batch. Please note that the timing includes both the process of visual feature extraction and the process of video summarization through our ptr-generator. During inference, we do not need to run the discriminator.

We can see that, for Youtube dataset, summarizing a video with 3.95 minutes long takes only 11.69 seconds via our proposed model. If we concatenate the 16 videos and perform summarization for them simultaneously, the overall time is much less than doing them one by one because of the parallel acceleration of GPU. A similar result can be found for LoL dataset. The result shows that our method can summarize the video accurately and efficiently.

5.4. Qualitative Results

To better visualize the characteristics of our method, we present some predictions done by our method on the YouTube dataset in Fig. 5. It can be seen that although our method may not select exactly the same frame as the ground truth, our generated summarization has very good visual similarity with the ground truth and also looks reasonable.

6. Conclusion

We have presented a GAN-based training framework aiming to combine unsupervised and supervised video summarization settings. Our method contains an attention-based Ptr-Net as the generator to produce the summarization of an input video by predicting the cutting points of fragments. We construct a 3D CNN classifier as the discriminator to judge if the summarizer-generated summarizations are as authentic and of quality as human-created summarizations. The proposed method outperforms previous methods on SumMe, TVSum, YouTube, and LoL datasets by **1.5%** to **5.6%**. We show that our Ptr-Net generator does not suffer from the problem of unbalanced training-test length, and our discriminator can make good use of those unpaired summarizations without original videos to improve the generator.

References

[1] A. Agrawal, J. Lu, S. Antol, M. Mitchell, C. L. Zitnick, D. Batra, and D. Parikh. Vqa: Visual question answering. In *ICCV*, 2015.

[2] S. E. F. Avila, A. P. B. Lopes, A. Jr., and A. de Albuquerque Arajo. Vsumm: a mechanism designed to produce static video summaries and a novel evaluation method. In *PRL*, 2011.

[3] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

[4] Z. Cernekova, I. Pitas, and C. Nikou. Information theory-based shot cut/fade detection and video summarization. In *TCSVT*, 2006.

[5] Y. Cong, J. Yuan, and J. Luo. Towards scalable summarization of consumer videos via sparse dictionary selection. In *IEEE TMM*, 2012.

[6] Ejaz, Naveed, I. Mehmood, and S. W. Baik. Efficient visual attention based framework for extracting key frames from videos. In *EURASIP*, 2013.

[7] M. Fleischman, B. Roy, and D. Roy. Temporal feature induction for baseball highlight classification. In *ACMMA*, 2007.

[8] C.-Y. Fu, J. Lee, M. Bansal, and A. C. Berg. Video highlight prediction using audience chat reactions. In *EMNLP*, 2017.

[9] B. Gong, W. Chao, K. Grauman, and F. Sha. Diverse sequential subset selection for supervised video summarization. In *NIPS*, 2014.

[10] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial networks. In *NIPS*, 2014.

[11] M. Gygli, H. Grabner, H. Riemenschneider, and L. V. Gool. Creating summaries from user videos. In *ECCV*, 2014.

[12] K. He, X. Zhang, S. R. abd, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[13] S. Hochreiter and J. Schmidhuber. Long short-term memory. In *Neural Computation*, 1997.

[14] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. In *TPAMI*, 2013.

[15] Z. Ji, K. Xiong, Y. Pang, and X. Li. Video summarization with attention-based encoder-decoder networks. In *arXiv preprint arXiv:1708.09545*, 2017.

[16] A. Khosla, R. Hamid, C.-J. Lin, and N. Sundaresan. Large-scale video summarization using web-image priors. In *CVPR*, 2013.

[17] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *arXiv preprint arXiv:1412.6980*, 2014.

[18] S. K. Kuanar, R. Panda, and A. S. Chowdhury. Video key frame extraction through dynamic delaunay clustering with a structural constraint. In *JVCIR*, 2013.

[19] M. Lewis, D. Yarats, Y. Dauphin, D. Parikh, and D. Batra. Deal or no deal? end-to-end learning of negotiation dialogues. In *EMNLP*, 2017.

[20] Y. Ma, L. Lu, H. Zhang, and M. Li. A user attention model for video summarization. In *ACMMA*, 2002.

[21] B. Mahasseni, M. Lam, and S. Todorovic. Unsupervised video summarization with adversarial lstm networks. In *CVPR*, 2017.

[22] S. Mei, G. Guan, Z. Wang, M. He, and D. D. Feng. Video summarization via minimum sparse reconstruction. In *JPRR*, 2015.

[23] C. W. Ngo, Y. Ma, and H. Zhang. Video summarization and scene detection by graph modeling. In *TCSVT*, 2005.

[24] L. Nie, R. Hong, L. Zhang, D. T. Yingjie Xia, and N. Sebe. Perceptual attributes optimization for multivideo summarization. In *IEEE Trans. Cybern*, 2016.

[25] M. Otani, Y. Nakashima, E. Rahtu, J. Heikkil, and N. Yokoya. Video summarization using deep semantic features. In *ACCV*, 2016.

[26] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *ICLR*, 2017.

[27] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid. Category-specific video summarization. In *ECCV*, 2014.

[28] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes. Tvsum: summarizing web videos using titles. In *CVPR*, 2015.

[29] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[31] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In *NIPS*, 2015.

[32] S. Wang and J. Jiang. Machine comprehension using match-lstm and answer pointer. In *ICLR*, 2017.

[33] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, 1992.

[34] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. In *Neural Computation*, 1989.

[35] K. Xu, J. L. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. S. Zemel, and Y. Bengio. Show, attend and tell: neural image caption generation with visual attention. In *ICML*, 2015.

[36] X. Xu, T. M. Hospedales, and S. Gong. Discovery of shared semantic spaces for multiscene video query and summarization. In *TCSVT*, 2017.

[37] L. Xuelong, B. Zhao, and X. Lu. A general framework for edited video and raw video summarization. In *IEEE TIP*, 2017.

[38] L. Yu, W. Zhang, J. Wang, and Y. Yu. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, 2017.

[39] K. Zhang, W. Chao, F. Sha, and K. Grauman. Summary transfer: exemplar-based subset selection for video summarization. In *CVPR*, 2016.

[40] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Video summarization with long short-term memory. In *ECCV*, 2016.

[41] K. Zhang, K. Grauman, and F. Sha. Retrospective encoders for video summarization. In *ECCV*, 2018.