Escaping local minima in deep reinforcement learning for video summarization

Panagiota Alexoudi
Ioannis Mademlis
Ioannis Pitas
palexoud@csd.auth.gr
imademlis@csd.auth.gr
pitas@csd.auth.gr
Department of Informatics, Aristotle University of Thessaloniki
Thessaloniki, Greece

ABSTRACT

State-of-the-art deep neural unsupervised video summarization methods mostly fall under the adversarial reconstruction framework. This employs a Generative Adversarial Network (GAN) structure and Long Short-Term Memory (LSTM) autoencoders during its training stage. The typical result is a selector LSTM that sequentially receives video frame representations and outputs corresponding scalar importance factors, which are then used to select key-frames. This basic approach has been augmented with an additional Deep Reinforcement Learning (DRL) agent, trained using the Discriminator's output as a reward, which learns to optimize the selector's outputs. However, local minima are a well-known problem in DRL. Thus, this paper presents a novel regularizer for escaping local loss minima, in order to improve unsupervised key-frame extraction. It is an additive loss term employed during a second training phase, that rewards the difference of the neural agent's parameters from those of a previously found good solution. Thus, it encourages the training process to explore more aggressively the parameter space in order to discover a better local loss minimum. Evaluation performed on two public datasets shows considerable increases over the baseline and against the state-of-the-art.

CCS CONCEPTS

• **Computing methodologies → Reinforcement learning**; **Video summarization**; **Neural networks**.

KEYWORDS

video summarization, key-frame extraction, unsupervised learning, deep reinforcement learning

ACM Reference Format:
Panagiota Alexoudi, Ioannis Mademlis, and Ioannis Pitas. 2023. Escaping local minima in deep reinforcement learning for video summarization. In *International Conference on Multimedia Retrieval (ICMR '23), June 12–15, 2023, Thessaloniki, Greece.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3591106.3592288

1 INTRODUCTION

The abundance of available video content has made automated video summarization a powerful tool in a variety of applications. *Automated video summarization* creates brief video summaries by selecting the most important and crucial elements of the original video. These are the most significant video frames, or segments of consecutive video frames, displayed chronologically to provide a succinct summary. Video summaries can be either static ones, consisting of key-frames , or dynamic ones, containing key-segments ordered as a trailer/skim [13, 14]. This paper concerns key-frame extraction.

In recent years, Deep Neural Networks (DNNs) have dominated the video summarization literature. Initially, supervised learning approaches were developed, that required manual "ground-truth" summaries to be present during training [25]. Typically, they utilized Convolutional Neural Networks (CNNs) pretrained for whole-image classification to derive semantic vector representations for each video frame. Each of these representations is then given to the summarization DNN, which chooses the key-frames. Supervised methods, however, depend on rather dubious summary ground-truth, derived from subjective user annotations. Additionally, obtaining such manual summaries for large-scale video datasets is very labor/time-intensive and supervised training leads to difficulties in generalization.

The alternative of unsupervised DNN-based key-frame extraction typically relies on Generative Adversarial Networks (GANs) [5] and Long Short-Term Memory (LSTMs) networks, employed for temporally modeling the original video. The main idea is that the summary (i.e., the temporally ordered set of extracted key-frames) must be able to, in a sense, reconstruct the original full-length video and this reconstruction should be indistinguishable from the original video [10, 11]. An orthogonal approach is to employ reinforcement learning in order to train a neural agent that achieves a high reward, properly defined for key-frame extraction. For instance, a combination of reward terms is proposed in [26], which encourages diversity and representativeness of the generated summaries. Diversity is measured by computing the pairwise dot products of video frame descriptors, while representativeness is measured by selecting video frames close to cluster centers in the video embedding space. In any case, the derived summary should simultaneously be representative of the original video content, concise and visually diverse.

The state-of-the-art in unsupervised key-frame extraction is the adversarial reconstruction framework [15]. It is composed of two main components: the *Summarizer* and the *Discriminator*. The Summarizer contains the *Selector*, the *Encoder* and the *Decoder*. It serves the role of the *Generator*, constructing training data points for the Discriminator (which is a binary classifier), under a GAN setting. These interacting components are LSTM networks and are trained concurrently, using back-propagation and any variant of gradient descent. The Selector generates importance scores that indicate each video frame's appropriateness for inclusion in the summary. Accordingly, given the chosen key-frames, the Autoencoder (Encoder-Decoder) tries to reconstruct the entire, original input video sequence, whilst the Discriminator is trained to distinguish between summary-based reconstructions and original videos. After training, the only component necessary to produce the summary of a new video is the Selector. This fundamental approach concentrates on the ability of the summary to recreate the initial video, but [15] also integrated a Determinantal Point Process (DPP) regularizer [12] during training, in order to obtain more visually diverse key-frames.

Various algorithms have built upon the original method from [15], such as [3], [2] and [1]. The approach most relevant to this paper is [1], which embedded a DRL Actor-Critic agent into the training process. The Actor receives as its initial input state the State Generator's output, i.e., the vector of scalar importance scores for all original video fragments (non-overlapping segments of multiple consecutive video frames), and gradually modifies it; these modifications stem from the actions performed by the agent. The Discriminator's output is exploited as a reward guiding this DRL task: essentially, Actors which construct summaries that maximally confuse the Discriminator are rewarded. As in all Actor-Critic DRL methods, a separate Critic neural model evaluates the Actor's choices during the training stage, in order to facilitate learning of the optimal policy/mapping between states and actions. This mapping is encoded in the Actor's learnable parameters. After training has been completed, the Actor and the Selector are the only neural modules required for key-frame extraction in new, test videos. They form a pipeline, with the Actor transforming the output of the Selector into an optimized set of importance scores.

Although this DRL-enhanced variant of the adversarial reconstruction framework has led to state-of-the-art results in unsupervised key-frame extraction, it is well-known that DRL may suffer from entrapment in suboptimal local loss minima [19]. Thus, this paper proposes a new regularizer for escaping local minima, under the guise of a novel loss term introduced into the training process of the Actor-Critic model in any DRL-based baseline method for key-frame extraction. Adding this regularizer during training augments the quality of the DRL agent by compelling it to escape the local minimum it normally tends to converge during training, thus allowing its optimization to reach a better solution. This regularizer may easily be added to the pool of loss functions used for training the overall framework, with its gradient signal specifically influencing the Actor-Critic module. Notably, it is entirely different from common ways of modifying the DRL learning objective for achieving increased exploration during training (e.g., by policy entropy maximization in Soft Actor-Critic [7]).

A quantitative assessment using common protocols on two publicly available datasets, TVSum and SumMe, reveals favorable findings and non-negligible increases over the baseline.

2 ESCAPING LOCAL MINIMA

The proposed method ($\mathcal{L}_{elm}$) is a potential addition to any DRL-based deep neural key-frame extraction method that relies on Actor-Critic agents. It is a training-stage regularizer that can be added to the original pool of loss functions influencing the optimization of the Actor-Critic models. It operates by segmenting training into two phases. The first phase is exactly identical to the complete baseline method's training stage, proceeding for K epochs without $\mathcal{L}_{elm}$. During the second phase, the final trained baseline Actor-Critic model/agent from the K-th epoch of the first phase is exploited as a reference "frozen" model. This second training phase proceeds for another K epochs, but this time with the proposed loss term $\mathcal{L}_{elm}$ punishing at each iteration the similarity between the reference frozen Actor-Critic model and the current one. This similarity is computed within $\mathcal{L}_{elm}$ in terms of the model parameters. This additional optimization objective forces the agent's training process to search for a different local loss minimum than the one found during the traditional first phase (the first K epochs, without the $\mathcal{L}_{elm}$ regularizer).

This integrated compulsion towards diversity in the parametric structure of the agent, i.e., the difference between the final solution and the previously found reference model, leads at the end of the second training phase to an Actor with better summarization performance. This comes at zero overhead in terms of inference runtime during the test stage. The obvious drawback of an approximately double required time interval compared to baseline (since the baseline architecture is trained for K epochs, i.e., the first phase, while the proposed method needs training for $2K$ epochs, i.e., the first and the second phase) is irrelevant to the actual deployment of a pretrained summarization model.

The proposed method can be applied generally, as an add-on to any DRL-based deep neural key-frame extraction framework, but was actually implemented and evaluated on top of a DRL-enhanced version of the adversarial reconstruction framework [15], namely AC-SUM-GAN [1]. *This choice was made simply because AC-SUM-GAN is currently at the state-of-the-art for unsupervised key-frame extraction. In principle, however, the proposed method can be identically applied to any other DRL-based deep neural key-frame extraction framework.* The baseline AC-SUM-GAN is briefly described below.

2.1 Baseline Framework

Architecture. The original T video frames of the full-length sequence are assumed to be represented by a corresponding set of T temporally ordered convolutional feature description vectors $\mathbf{X} \in \mathbb{R}^{P \times T}$, derived by a Convolutional Neural Network (CNN) pretrained for whole-image classification on a large-scale image dataset. These vectors (the columns of $\mathbf{X}$) are sequentially fed to a Linear Compression layer that reduces their size by half and, thus, sequentially outputs the columns of the matrix $\mathbf{X}' \in \mathbb{R}^{\frac{P}{2} \times T}$. This is the set of video frame descriptions which, from now on, will represent the video to be summarized. After the Linear Compression layer, the basic training-stage architectural components are the

State Generator (bi-directional LSTM), the *Actor* and the *Critic* (two fully connected networks), the *Fragment Selector* (matrix multiplication operator), the *Variational Auto-Encoder* (VAE) (two LSTMs) and the *Discriminator* (LSTM).

During training, the State Generator (equivalent to the Selector in [15]) sequentially receives as its input the compressed feature vectors $\mathbf{x'}_{*i}$, $1 \leq i \leq T$ and assigns to them video frame-level scalar importance scores: $\mathbf{s} \in \mathbb{R}^T$. Thus, it produces the initial input state for the DRL agent, in the form of the vector $\mathbf{f} \in \mathbb{R}^M$ which contains scores at a coarser, segment-level, by averaging the importance scores of all video frames within each fragment:

$$f_j = (\sum_{t=(j-1)d+1}^{jd} s_t)/d. \quad (1)$$

Here, M defines the number of non-overlapping and consecutive "fragments" into which the original video is automatically segmented, while $d \in \mathbb{N}$ is the duration of each video fragment in video frames. The Actor and the Critic receive $\mathbf{f}$ as a current input state and play an "N-picks" game. At the i-th iteration of this game, where $i \in 1, 2, ..., N$, the Critic produces a scalar value v_i, which acts as an assessment for the Actor's choices (typical in all Actor-Critic DRL methods). At each iteration of the game, the Actor produces a distribution of actions $\mathbf{c}_i \in \mathbb{R}^M$ and updates/modifies the video frame-level score vector $\mathbf{s}$ by sampling this distribution. Each action selects one new video fragment for inclusion into the summary, leading to an increase in the scalar importance scores of video frames that belong to the selected fragments, at the expense of the score of the other video frames.

The Fragment Selector utilizes the modified importance scores $\mathbf{s'}$ to multiply each of the compressed feature vectors in $\mathbf{X'}$ with its respective score. These scaled video frame representations essentially form the currently extracted summary and are fed to the Variational Autoencoder, which is tasked to sequentially reconstruct the original video frames as the columns of matrix $\hat{\mathbf{X}} \in \mathbb{R}^{P \times T}$. Finally, the Discriminator receives as separate training inputs the matrix $\mathbf{X'}$ (original video frame representations) and the matrix $\hat{\mathbf{X}}$ (summary-based reconstruction of the full-length video frame representations), returning a reward to the Critic.

Loss functions. The loss functions utilized during training are described below. The *Reconstruction Loss* computes the difference in the proximity of the original and reconstructed feature vectors, $\mathcal{L}_{recon} = ||\phi(\mathbf{X}) - \phi(\hat{\mathbf{X}})||^2$, where $\phi(.)$ is the last internal LSTM state of the Discriminator. The *Prior Loss* ($\mathcal{L}_{prior} = D_{KL}(q(e|x)||p(e))$) estimates the amount of information lost when the Encoder's latent space is employed to describe the VAE's previous distribution. The *Sparsity Loss* $\mathcal{L}_{sparsity} = ||\frac{1}{M}\sum_{t=1}^{M} s_t - \sigma||_2$, tries to compel the State Generator to create sparse importance scores, according to the scalar hyperparameter $\sigma \in [0, 1]$, so that only a small percentage of the original video frames (adjusted by σ) are labeled key-frames in the end.

The Original Video Loss $\mathcal{L}_{orig} = (1 - p(\mathbf{X}))^2$, pushes towards minimization of the distance between the original video class label (the number 1) and the computed probability of an original video at the Discriminator's output, when the latter one is indeed fed an original video. Accordingly, the Summary Loss $\mathcal{L}_{sum} = \left(p(\hat{\mathbf{X}})\right)^2$, tries to minimize the distance between the summary video class label (the number 0) and the computed probability at the Discriminator's output of a summary-based reconstruction, when indeed a reconstruction is fed to the classifier. Jointly, these two supervised loss terms force the Discriminator to become a binary classifier. On the other hand, the Generator Loss $\mathcal{L}_{gen} = \left(1 - p(\hat{\mathbf{X}})\right)^2$ tries to minimize the distance between the original video class label and the computed probability of an original video at the Discriminator's output, when the latter one is fed a reconstructed video as input. So, this is analogous to the adversarial loss of a typical GAN, aiming to compel the Summarizer to derive a summary-based reconstruction indistinguishable from an original video.

The Critic Loss $\mathcal{L}_{critic} = \frac{1}{N}\sum_{i=1}^{N} \alpha_i^2$ minimizes the advantage a_i, as it computes the value v_i that is used to evaluate the Actor's choices during the N iterations of the "N-picks" game. The advantage is computed as $a_i = z_i - v_i$, $i \in [1, N]$ and shows the superiority of a given action in contrast to the average action in a specific state of the N-picks game. The return is $z_i = \sum_{k=i}^{N} \gamma^{k-i} r_i$, with $\gamma \in [0, 1]$ being a discount factor and r_i the reward provided by the Discriminator. This reward is simply $1 - \mathcal{L}_{recon}$. After the i-th step of the "N-picks" game, the Actor receives a feedback from the Critic. The Actor Loss is $\mathcal{L}_{actor} = -\frac{1}{N}(\sum_{i=1}^{N} \ln \mathbf{c_i} a_i + \delta \sum_{i=1}^{N} H(\mathbf{c_i}))$, where δ is an entropy regularizer, $\mathbf{LP} = \ln\{\mathbf{c_i}\}_{i=1}^{N}$ is the logarithm of the probability density function c_i and $E_n = \sum_{i=1}^{N} H(\mathbf{c_i})$ is the entropy of c_i. The Actor tries to learn a policy that optimizes the likelihood of a significant fragment being included in the summary.

The above loss functions are computed incrementally during the training of the model. The algorithm computes $\mathcal{L}_{prior}$ and $\mathcal{L}_{recon}$ in the first forward pass; during the backward pass it updates the Encoder. Then, during the second forward pass, it computes $\mathcal{L}_{gen}$ and $\mathcal{L}_{recon}$, while subsequently it updates the Decoder. The reconstructed feature vectors $\hat{\mathbf{X}}$ are produced and utilized to compute $\mathcal{L}_{sum}$, during the third forward pass. Thereafter, the compressed feature vectors $\mathbf{X'}$ are used as training input for the Discriminator and the $\mathcal{L}_{orig}$ term is computed. The sum of these two losses in then used to update the Discriminator and the Linear Compressor.

In the fourth and final step, the Actor-Critic model plays the "N-picks" game, producing the value v, the reward r, $\mathbf{LP}$ and E_n. These values are utilized to compute $\mathcal{L}_{actor}$, $\mathcal{L}_{critic}$ and $\mathcal{L}_{sparsity}$ and train the Critic, the Actor, the State Generator and the Linear Compressor.

Test. During the test stage, after training has been completed, only the State Generator and the Actor are required. The summary of an input video is constructed using the scores of vector $\mathbf{s}$, after it has been modified by the Actor. Key-frame selection is performed in the following manner: a) Kernel Temporal Segmentation (KTS) [16] automatically segments the original video into consecutive shots of different duration, based on an analysis of $\mathbf{X}$, in order to assign a weight to each video frame (equal to the total number of video frames in its shot), and b) the importance score and the weight of each video frame are exploited to finally select the key-frame set by solving the combinatorial knapsack problem.

2.2 Proposed Method (ELM Loss)

The proposed regularizer $\mathcal{L}_{elm}$ can be applied by doubling the number of epochs the baseline DRL-based summarization model is trained. During the initial K epochs, training proceeds as usual. At the end of the K-th epoch, the parameters of the final trained baseline Actor and Critic models are stored as two reference vectors. Subsequently, training resumes with an identical copy of the neural architecture and proceeds for a second phase of K epochs. The only difference from the first phase is that the proposed ELM Loss term is subtracted from the computed Actor Loss and Critic Loss. Thus, during the second phase:

$$\mathcal{L}_{actor,elm} = \mathcal{L}_{actor} - \lambda \mathcal{L}_{elmA} \tag{2}$$

and

$$\mathcal{L}_{critic,elm} = \mathcal{L}_{critic} - \lambda \mathcal{L}_{elmC}, \tag{3}$$

where $\mathcal{L}_{actor,elm}/\mathcal{L}_{critic,elm}$ is the loss function updating the Actor/Critic, respectively, during the novel second training phase, while $\mathcal{L}_{elmA}/\mathcal{L}_{elmC}$ is the version of the proposed regularizer for updating the Actor/Critic, respectively. $\lambda > 0$ is a coefficient adjusting how much the proposed loss term is taken into account by the optimization process, against the task-specific $\mathcal{L}_{actor}$ and $\mathcal{L}_{critic}$ loss terms.

$\mathcal{L}_{elm}$ is computed as the distance between the parameter vector of the current training iteration's agent during the on-going second phase from the respective stored/frozen parameter vector of the reference baseline agent (obtained previously, at the end of the first training phase). This is done separately for the Actor and for the Critic, resulting in the differentiation between $\mathcal{L}_{elmA}$ and $\mathcal{L}_{elmC}$. Such a distance can be calculated individually for each of the agent's neural layers; these partial distances can then be summed to form the loss value. Below, the difference between $\mathcal{L}_{elmA}$ and $\mathcal{L}_{elmC}$ is ignored for purposes of clearer presentation, since they only deviate to one another with respect to where the stored reference parameter vector came from (the reference Actor/Critic, correspondingly). Thus, the following general definition of $\mathcal{L}_{elm}$ holds:

$$\mathcal{L}_{elm} = \sum_{i=1}^{n} \left(1 - S_C(\mathbf{w}_i^C, \mathbf{w}_i^R)\right), \tag{4}$$

where S_C is the cosine similarity between vectors, $\mathbf{w}_i^C/\mathbf{w}_i^R$ is the parameter vector of the i-th layer of the current/reference agent, respectively, and n is the number of layers in the agent. This formulation does not penalize an agent that is identical to the reference one, but essentially rewards (by reducing the loss value in Eqs. (2) and (3)) one that is different from the reference one in terms of cosine distance. As previously noted, Eq. (4) is obviously implemented differently for the Actor and for the Critic, since these two agents correspond to different stored reference parameter vectors.

The hyperparameter λ cannot be kept large during the entire second phase. It needs to be rather high during the initial epochs of the second phase, but it should be reduced gradually in order to eventually allow the task-specific loss terms to take over the optimization process, through their gradient signals. Ideally, training with $\mathcal{L}_{elmA}/\mathcal{L}_{elmC}$ and a large λ during the initial epochs of the second phase will ultimately lead the optimization process towards novel regions of the agent's parameter space, i.e., areas not reached during the first phase, and therefore increase the chance of subsequently discovering a local loss minimum which is better than that of the reference agent. But for actually reaching such a good task-specific loss minimum near the end of the second phase, λ must be small during the latter epochs.

Method	*TVSum*	*SumMe*
Online Motion-AE [23]	51.5%	37.7%
SUM-FCN$_{unsup}$ [18]	52.7%	41.5%
DR-DSN [26]	57.6%	41.4%
EDSN [4]	57.3%	42.6%
Unpaired VSN [17]	55.6%	47.5%
PCDL [24]	58.4%	42.7%
ACGAN [8]	58.5%	46.0%
SUM-GAN-sl [3]	58.4%	47.8%
SUM-GAN-AAE [2]	58.3%	48.9%
CSNet [9]	58.8%	51.3%
AC-SUM-GAN [1]	60.6%	50.8%
AC-SUM-GAN (200 epochs)	61.4%	54.4%
Proposed ([1] - $\lambda\mathcal{L}_{elm}$)	**62.0%**	**55.8%**

Table 1: Comparison of various deep unsupervised video summarization methods on the TVSum and SumMe datasets, using the F-score metric (percentage, higher is better). Best results are in bold.

3 EVALUATION

The proposed method was implemented on top of AC-SUM-GAN and evaluated using two publicly available datasets: TVSum [20] and SumMe [6]. SumMe contains 25 videos of duration 1 to 6 minutes, while TVSum consists of 50 videos ranging from 1 to 11 minutes. Both datasets include single-user ground-truth summary for every video and manual annotations for key-fragments (SumMe) or video frame-level importance scores (TVSum). The protocol used for evaluation is the key-fragment-based approach with the F-score metric [22], as executed in [1]. The dataset was divided into 5 random splits, while 80% of the videos were used for training and 20% for testing.

Videos were downsampled at 2 FPS and video frame representations were derived from the pool5 layer of a GoogLeNet CNN pretrained for whole-image classification on the large-scale ImageNet dataset. The derived per-frame feature vector dimensionality is decreased from $P = 1024$ to 512 by the Linear Compression layer. The Encoder, Decoder and Discriminator consist of two LSTM layers, whereas the State Generator is a bi-directional LSTM. The Actor contains 4 fully connected layers. Its output is fed to a softmax layer and produces a categorical distribution of probabilities, while the Critic contains 5 fully connected layers and its output is a scalar value ranging from 0 to 1. The Adam optimizer was used during training, with a learning rate of 10^{-4} for every component besides the Discriminator, for which it was set to 10^{-5}. σ was set to 0.4 for SumMe and 0.9 for TVSum.

The initial parameters of both the Actor and the Critic were set using Dirac initialization [21]. Training hyperparameter λ was empirically set to an initial value of 2/1 with an exponential/constant decay over the epochs of the second phase, for SumMe/TVSum respectively. The constant decay was set to a step of 0.1 every 10 epochs.

The proposed training-stage method, implemented on top of the baseline AC-SUM-GAN, is compared in terms of summarization performance (measured in F-Score) against several unsupervised key-frame extraction approaches in Table 1. To achieve a fair comparison, the performance of the baseline AC-SUM-GAN is reported not only for 100 epochs (as in [1]), but also for 200 epochs, since the proposed method requires double the typical number of training epochs. As it can be seen, adding $\mathcal{L}_{elm}$ to the pool of loss functions leads to non-negligible test-stage gains in F-Score with regard to the directly comparable AC-SUM-GAN-200-epochs competitor, which is the second best performer.

4 CONCLUSIONS

A novel regularizer for escaping local loss minima was presented, suitable for deep reinforcement learning (DRL)-based unsupervised key-frame extraction methods relying on an Actor-Critic module. Adding this regularizer to the pool of employed loss functions during the training stage leads the optimization process towards novel regions of the agent's parameter space, i.e., areas not typically reached. Therefore, it increases the chance of subsequently discovering a local loss minimum which is better than that of the baseline model, when trained without the proposed regularizer. Although this requires double the training epochs compared to typical approaches, it induces zero runtime overhead during the test stage. The method was implemented on top of a state-of-the-art DRL-enhanced variant of the common adversarial reconstruction framework and evaluated on two publicly available datasets. Results indicate non-negligible gains compared to baseline.

ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 951911 (AI4Media). This publication reflects only the authors' views. The European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] E. Apostolidis, E. Adamantidou, A. I. Metsai, V. Mezaris, and I. Patras. 2020. AC-SUM-GAN: Connecting actor-critic and generative adversarial networks for unsupervised video summarization. *IEEE Transactions on Circuits and Systems for Video Technology* 31, 8 (2020), 3278–3292.

[2] E. Apostolidis, E. Adamantidou, A. I. Metsai, V. Mezaris, and I. Patras. 2020. Unsupervised video summarization via attention-driven adversarial learning. In *International Conference on Multimedia Modeling (MMM)*. Springer.

[3] E. Apostolidis, A. I. Metsai, E. Adamantidou, V. Mezaris, and I. Patras. 2019. A stepwise, label-based approach for improving the adversarial training in unsupervised video summarization. In *Proceedings of the International Workshop on AI for Smart TV Content Production, Access and Delivery*.

[4] N. Gonuguntla, B. Mandal, and NB Puhan. 2019. Enhanced Deep Video Summarization Network. In *Proceedings of the British Machine Vision Conference (BMVC)*.

[5] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y.A. Bengio. 2014. Generative adversarial nets. *Proceedings of the Advances in Neural Information Processing Systems (NIPS)* (2014).

[6] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool. 2014. Creating summaries from user videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer.

[7] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the International Conference on Machine Learning*. PMLR.

[8] X. He, Y. Hua, T. Song, Z. Zhang, Z. Xue, R. Ma, N. Robertson, and H. Guan. 2019. Unsupervised video summarization with attentive conditional Generative Adversarial Networks. In *Proceedings of the ACM International Conference on Multimedia*.

[9] Y. Jung, D. Cho, D. Kim, S. Woo, and I. S. Kweon. 2019. Discriminative feature learning for unsupervised video summarization. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.

[10] M. Kaseris, I. Mademlis, and I. Pitas. 2021. Adversarial unsupervised video summarization augmented with dictionary loss. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*.

[11] M. Kaseris, I. Mademlis, and I. Pitas. 2022. Exploiting Caption Diversity for Unsupervised Video Summarization. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[12] A. Kulesza and B. Taskar. 2012. Determinantal Point Processes for machine learning. *arXiv preprint arXiv:1207.6083* (2012).

[13] I. Mademlis, A. Tefas, N. Nikolaidis, and I. Pitas. 2016. Movie shot selection preserving narrative properties. In *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*.

[14] I. Mademlis, A. Tefas, N. Nikolaidis, and I. Pitas. 2016. Multimodal stereoscopic movie summarization conforming to narrative characteristics. *IEEE Transactions on Image Processing* 25, 12 (2016), 5828–5840.

[15] B. Mahasseni, M. Lam, and S. Todorovic. 2017. Unsupervised video summarization with adversarial lstm networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[16] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid. 2014. Category-specific video summarization. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer.

[17] M. Rochan and Y. Wang. 2019. Video summarization by learning from unpaired data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[18] Mrigank Rochan, Linwei Ye, and Yang Wang. 2018. Video summarization using fully convolutional sequence networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer.

[19] M. Schilling, A. Melnik, F. W. Ohl, H. J. Ritter, and B. Hammer. 2021. Decentralized control and local information for robust and adaptive decentralized Deep Reinforcement Learning. *Neural Networks* 144 (2021), 699–725.

[20] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes. 2015. TVSum: Summarizing web videos using titles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[21] S. Zagoruyko and N. Komodakis. 2017. DiracNets: Training very deep neural networks without skip-connections. *arXiv preprint arXiv:1706.00388* (2017).

[22] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. 2016. Video summarization with Long Short-Term Memory. In *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer.

[23] Y. Zhang, X. Liang, D. Zhang, M. Tan, and E. P. Xing. 2020. Unsupervised object-level video summarization with online motion auto-encoder. *Pattern Recognition Letters* 130 (2020), 376–385.

[24] B. Zhao, X. Li, and X. Lu. 2019. Property-constrained dual learning for video summarization. *IEEE Transactions on Neural Networks and Learning Systems* 31, 10 (2019), 3989–4000.

[25] B. Zhao, X. Li, and X. Lu. 2020. TTH-RNN: Tensor-Train hierarchical recurrent neural network for video summarization. *IEEE Transactions on Industrial Electronics* (2020).

[26] K. Zhou, Y. Qiao, and T. Xiang. 2018. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*.