ATTN-SUM-GAN: Attentive Network for Unsupervised Video Summarization with Generative Adversarial Network

1st Minh-Nam Tran
University of Science
Viet Nam National University Ho Chi Minh City
Ho Chi Minh, Vietnam
tmnam20@apcs.fitus.edu.vn

2nd Viet-Nhat Thai
University of Science
Viet Nam National University Ho Chi Minh City
Ho Chi Minh, Vietnam
tvnhat20@apcs.fitus.edu.vn

***Abstract*—Video summarization is the task of selecting a subset of frames in the video. This study addresses the limitations of existing methods, particularly their slow computational speed and inability to capture long-range dependencies due to the usage of recurrent neural networks (RNN). To overcome these issues, the study introduces the ATTN-SUM-GAN model, replacing RNN units with scaled-dot product attention modules from the Transformer block. The Attentive modules, including selector, encoder, decoder, and discriminator, demonstrate superior performance on benchmark datasets (TVSum). The proposed model outperforms state-of-the-art methods on TVSum, achieving an average F1 score of 65.4%. The ablation study on hyperparameter configurations reveals optimal settings for this task, such as a hidden size of 64, further enhancing model performance. Contributions include the introduction of the Attentive modules, surpassing existing state-of-the-art methods on the TVSum dataset, and an ablation study on hyperparameters. The proposed ATTN-SUM-GAN model represents a significant advancement in video summarization, offering improved efficiency and performance.** [1]

***Index Terms*—video summarization, unsupervised learning, generative adversarial network, attentive network, transformer**

I. Introduction

In the 2020s, the volume of video has been observed to increase exponentially, especially on social media platforms [1]. It leads to the issues of storing, managing, retrieving, and sharing videos [2]. Therefore, an emerging need for fast, efficient video summarization has been raised. Several methods with deep learning models are introduced as solutions to this problem, including supervised methods, unsupervised methods [3], [4], [5], reinforcement learning [6], [7], [8], and contrastive learning [9]. However, such methods are struggling with their slow computational speed and weak ability to capture long-range dependencies due to the usage of recurrent neural networks (RNN) [10] in their models.

Several methods employ the attention architecture to solve the long-range dependency issue. However, the slow computational speed is the nature of the RNN family [10], [11], [12]. Therefore, this study aims to replace the use of recurrent neural networks to improve computational speed as well as enhance the ability to capture long-distance dependency.

To tackle the mentioned phenomena, the recurrent neural network units are switched out for the scaled-dot product attention module in the Transformer block [13] to fix the issues listed above. The Attentive modules, which include the selector, encoder, decoder, and discriminator, are built on top of the Transformer block [13]. This makes the model perform better on two benchmark datasets, SumMe [14] and TVSum [15], while cutting the training time by up to 18 times in the small configuration and 13 times in the base configuration on the TVSum dataset (see Figure 1). The proposed model, ATTN-SUM-GAN, outperforms current state-of-the-art methods on the TVSum dataset [15], with the average F1 score at 64.63%.

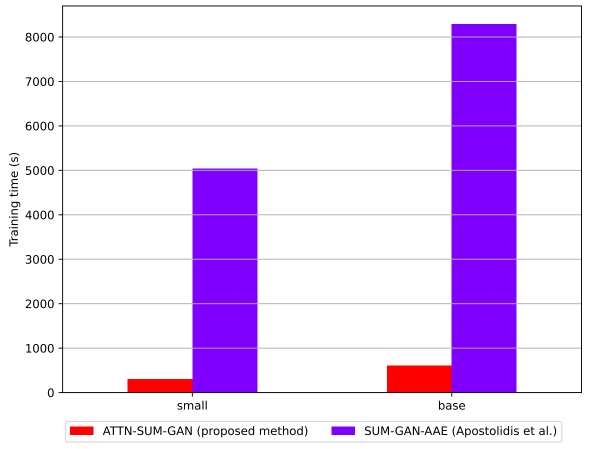


Fig. 1. Training time of proposed method and SUM-GAN-AAE [16] on TVSum dataset [15] with two configurations: small and base.

Besides the model improvement, empirical studies on the hyperparameters of model configurations are also examined, including the hidden sizes, the number of heads, and the number of layers. The evaluation results indicate that with

This work is completed during the course CS412 - Computer Vision.
[1]The code is available at https://github.com/trminhnam/ATTN-SUM-GAN

a hidden size of 64, the proposed architecture achieved the highest performance on the TVSum dataset [15].

In conclusion, the contributions are as follows:

- Proposing the Attentive modules that get rid of the recurrent neural networks by using Transformer block [13].
- The proposed network outperforms all state-of-the-art methods on the TVSum dataset [15].
- Ablation study about the hyperparameters of the Attentive block configuration is examined.

The structure of the paper has five sections. Section I introduces the work in general, followed by Section II giving information about previous works on video summarization. Next, Section III proposes the new method with ATTN-SUM-GAN architecture and the way to integrate the Transformer block [13] into the GAN modules [17]. The results and ablation study are reported in Section IV and Section V, respectively.

II. Related Works

Video summarization is the task of finding a shorter sequence of frames that represents a lengthy video [2]. The summary includes the critical and relevant data from the video clip in condensed form [18], [19].

Before 2015, scientists used hand-crafted patterns to pick out the important frames by looking at the scores of heuristic functions [20], [21], [22], [15]. As computer hardware has improved quickly, more scientists are interested in deep-learning-based methods that use neural networks, which provide more accurate results. The convolutional neural networks are used to extract features, while the recurrent networks are responsible for finding the relationships between video frames [23], [24].

Based on the type of training dataset, deep-learning-based video summarization techniques are categorized as supervised and unsupervised. The supervised methods [25] require the annotation of keyframes in the training video samples and tries to minimize the loss of the summarizer concerning the labels [26]. On the other hand, the unsupervised methods [5] use unlabeled data, which does not require much human effort to annotate the training set [27]. Since a wide range of video domains makes providing reliable and accurate large amounts of labeled datasets impossible, the unsupervised method is considered a common solution. To make the networks learn from unsupervised datasets, several training frameworks are effectively facilitated, including the reinforcement learning (RL) algorithm [6], the generative adversarial network (GAN)[17], and the contrastive learning method [9].

Several video summarization methods use generative adversarial networks (GAN) [24], [28], [29]. In 2017, Mahasseni et al. introduced SUM-GAN [24], which contains two components: the summarizer and the discriminator. The summarizer is the autoencoder [30] LSTM [11] network focusing on selecting keyframes, encoding, and decoding the selected frames for video reconstruction in the GAN framework, while the discriminator is responsible for distinguishing between the original and the synthesized videos [24]. In 2019, Jung et al. proposed the Chunk and Stride Network (CSNet) [28] to address the issue of long video by using local chunks and global strides to effectively capture video features. In the meantime, Yuan et al.'s Cycle-SUM [27] uses bi-directional cycle-consistent LSTM to maximize the information in the summary. Apostolidis et al. suggested SUM-GAN-AAE [16] in 2020 as a way to improve the generator in the GAN framework by integrating the attention mechanism into the variational autoencoder. In 2021, Apostolidis et al. continued to propose AC-SUM-GAN [31], an architecture that integrates the actor-critic model [32] for incrementally selecting video frames in the summary. SUM-GAN-GEA [33], introduced by Yu et al. in 2022, uses a graph model to extract the global features to select the keyframes for the summary.

Besides the GAN series for unsupervised video summarization, several studies employ the reinforcement learning algorithm [8], [3], [34], [7]. In 2018, Zhou et al. proposed DSN [8], a deep summarization network built on CNN [35] and bidirectional LSTM [11]. The technique considers the input video, which is basically a sequence of frames, as a consecutive collection of binary decision problems and models that each frame may be chosen or not to appear in the summary. The DSN [8] is trained under the reinforcement learning algorithm with the help of the reward function to determine the diversity level and the meaning of the summaries. Yaliniz et al. use IndRNN [36], a different kind of recurrent neural network [10], to lessen the impact of gradient vanishing [37]. This makes the networks scalable in terms of the number of layers and training steps. Li et al. use a smaller network for video topic classification and feature extraction, denoted as the video classification sub-network (VCSN) [34]. Then, the summarizer takes the representations from the VCSN to give the summary of the video [34]. In 2022, Liu et al. proposed 3DST-UNet-R [38], an approach that makes use of 3D spatial-temporal features via the UNet architecture [39] and trains the model with the reinforcement learning algorithm [6].

Recurrent neural networks (RNN) [10], long short-term memory (LSTM) [11], gated recurrent units (GRU) [12], and other types are used as building blocks for the aforementioned networks. However, the problems of the RNN family are struggling to fully capture long-range dependencies and slowing down the computing speed because recurrent units do not handle data in parallel. The former issue is tackled by using attention mechanisms [40], [29], [16], [41]. Apostolidis et al. use Luong attention [42] between the encoder and the decoder of the variational autoencoder [30] to capture long-range dependency more efficiently [16]. He et al. propose attentive conditional GAN [4] with three modules: feature descriptor, generator, and discriminator. Although the feature descriptor is built with self-attention modules, the generator and the discriminator also use bidirectional LSTM, which causes a bottleneck for this network. Liang introduced CAAN [29], which uses the self-attention mechanism in the generator to find the relationship between features from a fully convolutional sequence network. However, the CAAN's discriminator [29] is built with LSTM [11] resulting in slow computation.

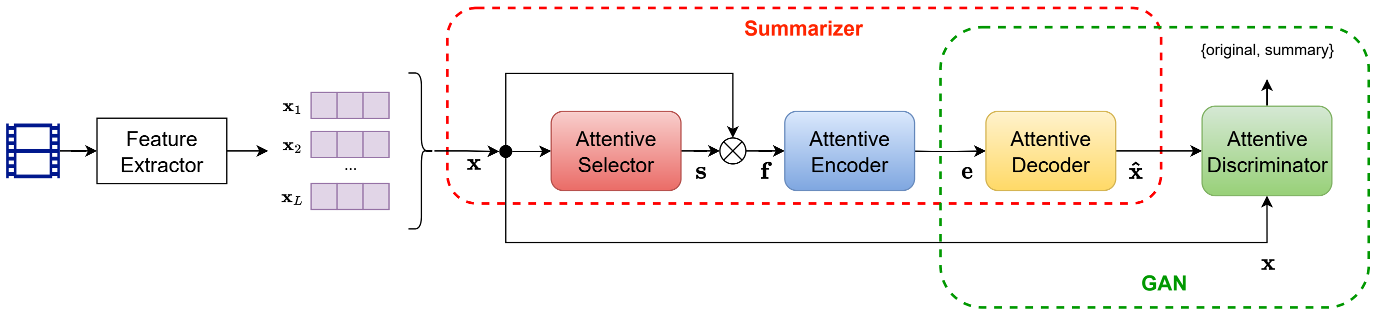


Fig. 2. Architecture of ATTN-SUM-GAN, which is similar to the SUM-GAN [24] and SUM-GAN-AAE [16]. The pipeline contains five modules, including the feature extractor (CNN), Attentive Selector, Attentive Encoder, Attentive Decoder, and Attentive Discriminator.

Since using recurrent neural networks slows down training and inference procedures due to their nature, the proposed method aimed to replace those recurrent units completely by the self-attention in Transformer block [13] proposed by Vaswani et al., resulting in decreasing training time and enhancing the ability to capture long-distance dependency.

III. Proposed Method

This section introduces the ATTN-SUM-GAN method by leveraging the Transformer block with the multi-head attention module to extract the relationship between video frames. The mathematical formulas of Attentive Block are introduced in Section III-A, while the ATTN-SUM-GAN network is described in Section III-B.

A. Attentive Modules with Transformer Architecture

For the video summarization problem, the Attentive Modules are built on top of the Transformer encoder block [13].

First, the input video feature, $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\} \in \mathbb{R}^{L\times d}$, is projected into three different spaces, including query ($\mathbf{Q}$), key ($\mathbf{K}$) and value ($\mathbf{V}$) using three parameter matrices $\mathbf{W}^Q \in \mathbb{R}^{d\times d_Q}$, $\mathbf{W}^K \in \mathbb{R}^{d\times d_K}$, $\mathbf{W}^V \in \mathbb{R}^{d\times d_V}$ (see Equation 1). For simplicity, it is chosen as $d_Q = d_K = d_V = d$. In the equation, L denotes the sequence length or number of frames in the video, and d denotes the hidden dimension size.

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \mathbf{K} = \mathbf{X}\mathbf{W}^K, \mathbf{V} = \mathbf{X}\mathbf{W}^V \tag{1}$$

Next, each of the projected $\mathbf{Q}$, $\mathbf{K}$, $\mathbf{V}$ matrices are split into h smaller subspaces along the hidden dimension d (see Equation 2), where h denotes the number of heads.

$$\begin{aligned}[\mathbf{Q}_1, \ldots, \mathbf{Q}_h] &= \mathrm{Split}(\mathbf{Q})\\ [\mathbf{K}_1, \ldots, \mathbf{K}_h] &= \mathrm{Split}(\mathbf{K})\\ [\mathbf{V}_1, \ldots, \mathbf{V}_h] &= \mathrm{Split}(\mathbf{V})\end{aligned} \tag{2}$$

For every triplet $\{\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i\}$, the scaled dot-product attention for head$_i$ ($i \in [1, h]$) is calculated as in Equation 3.

$$\mathrm{head}_i = \mathrm{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d_k}}\right)\mathbf{V}_i \quad \text{where } d_k = \frac{d}{h} \tag{3}$$

The output of the multi-head attention module is obtained by concatenating all the heads and then projected to the output space by matrix $\mathbf{W}^O \in \mathbb{R}^{d\times d}$ (see Equation 4).

$$\begin{aligned}\mathbf{O} &= \mathrm{MultiHeadAttention}(\mathbf{Q}, \mathbf{K}, \mathbf{V})\\ &= \mathrm{Concat}(\mathrm{head}_1, \ldots, \mathrm{head}_h)\mathbf{W}^O\end{aligned} \tag{4}$$

While calculating the attention scores, there is no non-linear transformation of the input. Therefore, feeding the dataflow through a FeedForward network plays as a non-linear transformation. In Equation 5, the input is transformed with two linear projection $\{(\mathbf{W}^U, \mathbf{b}^U), (\mathbf{W}^D, \mathbf{b}^D)\}$ and two non-linear activation functions $\{\mathrm{ActFn}_1, \mathrm{ActFn}_2\}$.

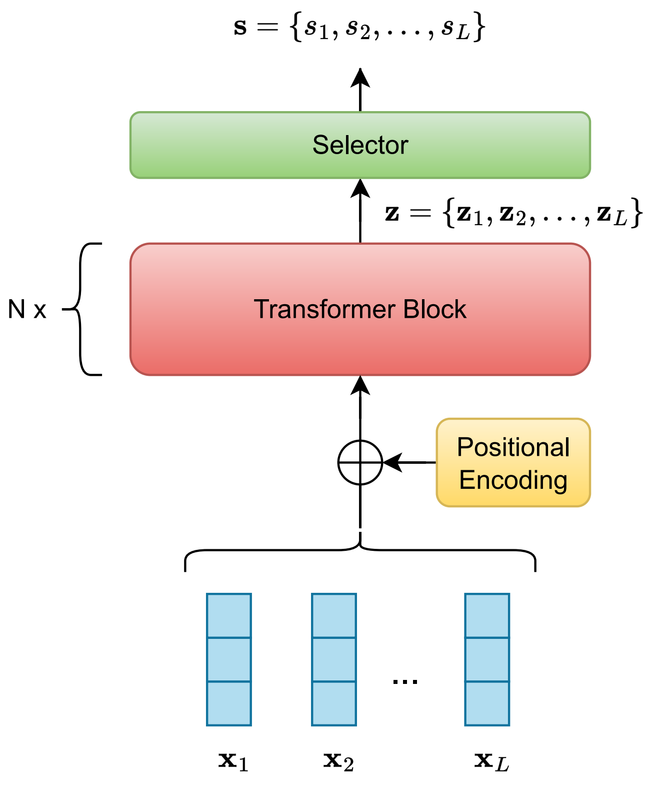
$$\begin{aligned}\mathbf{Z} &= \mathrm{FeedForward}(\mathbf{X})\\ &= \mathrm{ActFn}_1(\mathrm{ActFn}_2(\mathbf{X}\mathbf{W}_1 + \mathbf{b}^U)\mathbf{W}_2 + \mathbf{b}^D)\end{aligned} \tag{5}$$

Finally, the final formula for a Transformer block with layer normalization [43] (denoted as LayerNorm) and residual connection [44] after each module is described in Equation 6.

$$\begin{aligned}\mathbf{O} &= \mathrm{MultiHeadAttention}(\mathbf{X}\mathbf{W}^Q, \mathbf{X}\mathbf{W}^K, \mathbf{X}\mathbf{W}^V)\\ \mathbf{O'} &= \mathrm{LayerNorm}(\mathbf{O} + \mathbf{X})\\ \mathbf{Z} &= \mathrm{LayerNorm}(\mathbf{O'} + \mathrm{FeedForward}(\mathbf{O'}))\end{aligned} \tag{6}$$

In Figure 3, the Attentive Selector Module contains three submodules: the Positional Encoding, the Transformer Blocks, and the Selector. The input sequence $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L\} \in \mathbb{R}^{L\times d}$ combined with the positional encoding P as an element-wise addition. Then, it is passed through N Transformer Block to obtain the contextual representation $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_L\} \in \mathbb{R}^{L\times d}$. Finally, the Selector, containing a parameter matrix $\mathbf{W}^S$ and a Sigmoid activation function, gives an importance score for each frame from the corresponding representation $\mathbf{z}_i$.

The Attentive Discriminator is similar to the Attentive Selector but replaces the Selector Module with a mean reduction across the sequence length dimension from $\mathbb{R}^{L\times d} \to \mathbb{R}^d$, then passing the output through a fully connected layer and the sigmoid activation function to predict whether the input sequence is an original feature sequence or the synthesized one. The Attentive Encoder and Attentive Decoder are networks built on the same number of Transformer blocks.



$\mathbf{s} = \{s_1, s_2, \dots, s_L\}$
Selector
$\mathbf{z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_L\}$
N x
Transformer Block
Positional Encoding
...
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_L$

Fig. 3. Attentive Selector architecture. The input sequence of the video features is added with positional encoding, then passed through the N Transformer Block and the Selector at the end to get the importance scores.

B. ATTN-SUM-GAN

The pipeline contains two main components: the summarizer and the GAN [17] (visualized in Figure 2). The summarizer includes the Attentive Selector, the Attentive Encoder, and the Attentive Decoder, while the GAN framework has the Attentive Decoder and the Attentive Discriminator. In this architecture, all the modules except the CNN are built on top of the Transformer block [13] combined with a specific projection layer to map the dataflow into the hidden-size dimensional space. The problems of capturing long-term dependencies and bottleneck phenomena have been solved using the Attentive block instead of recurrent neural networks [10].

First, the features $\mathbf{X} \in \mathbb{R}^{L \times d}$ extracted from the video frame-by-frame via GoogLeNet [45] (L for the number of frames in the video, d for the hidden size, or the number of features). This step is done in the preprocessing stage.

Then, the captured sequence of features X is passed through the Attentive Selector to obtain the frame important scores $\mathbf{s} \in \mathbb{R}^{L \times 1}$ (see Figure 3), also known as the probability of a frame appearing in the summary. Then, input features $\mathbf{X}$ are multiplied point-wise with the corresponding important scores $\mathbf{s}$ to obtain the weighted features $\mathbf{w}$. After that, the Attentive Encoder receives $\mathbf{w}$ and encodes it into $\mathbf{e} \in \mathbb{R}^{L \times d}$.

Next is the dataflow inside the GAN framework [17]. The encoded features $\mathbf{e} \in \mathbb{R}^{L \times d}$ are then passed through the Attentive Decoder to reconstruct the feature sequences $\hat{\mathbf{X}} \in \mathbb{R}^{L \times d}$. The revamped sequence of features $\hat{X}$ and original features X are considered training samples of the Attentive Discriminator, where it is trained to recognize the real (original) and fake (generated) samples.

The loss function for this method is similar to the method of G. Liang et al. [29] and Apostolidis et al. [16], where there are three smaller components: adversarial loss, reconstruction loss, and sparsity loss. The adversarial loss makes the decoded weighted features $\mathbf{w}$ similar to the original features $\mathbf{X}$. The reconstruction loss minimizes the difference between the generated features $\hat{\mathbf{X}}$ and the original ones $\mathbf{X}$. Finally, the sparsity loss emphasizes that the model should focus on fewer frames in the selector.

IV. Experimental Results

In this section, experiments with the proposed architecture on the TVSum dataset are conducted. Besides testing the new architecture, the comparison with the baseline architecture, SUM-GAN-AAE [24], is also conducted with the same hyperparameter setting, as shown in Section IV-C.

A. Datasets

The performance of the proposed network is calculated on the TVSum [15] dataset.

The TVSum dataset [15] is a collection of 50 movies. Each movie has a duration ranging from 1 to 5 minutes. The dataset covers ten different categories from the TRECVid MED dataset. 20 users have annotated each movie using frame-level relevance scores and a single ground-truth summary created by averaging all the scores.

The dataset is split into five folds so that k-fold cross-validation is used to train and test the proposed method. This allows a pair comparison with the baseline [16].

B. Evaluation Metric

To compare the generated summary and the ground truth sequence, the key-fragment-based evaluation protocol [23] and F1 score (also known as F-score) are used.

The F-Score, also known as the F1 score, is a metric commonly used in binary classification to evaluate the performance of a model. It is the harmonic mean of precision $P = \dfrac{TP}{TP + FP}$ and recall $R = \dfrac{TP}{TP + FN}$ (see Equation 7).

$$F1 = \frac{2 \cdot P \cdot R}{P + R} \tag{7}$$

In this framework, the video summarization problem is shifted to a binary classification task where the model must decide whether each frame is included in the summary. Hence, using the F1 score as the evaluation metric is a suitable option.

C. Model Configurations and Hyperparameters

For training, the hyperparameters are listed as follows:

- Adam optimizer is used for training the whole process. The selector and generator use a learning rate of $1e-4$, while the discriminator is trained with a learning rate of $1e-5$. Other shared parameters are $\beta_1 = 0.9$, $\beta_2 = 0.999$, and the regularization factor of 0.15.
- Each training session runs the models through 100 epochs, and the gradients are clipped to have a maximum value of 5.0, avoiding gradient exploding.

The model configuration and the corresponding number of trainable parameters (denoted as #params) are shown in Table I. With the same number of layers and hidden size, ATTN-SUM-GAN architecture has fewer training parameters with respect to the SUM-GAN-AAE method .

TABLE I
MODEL CONFIGURATION FOR SUM-GAN-AAE [16] AND ATTN-SUM-GAN (PROPOSED) FOR FAIR COMPARISON.

Model name	**#layers**	**d**	**#heads**	$\mathbf{d_{ff}}$	**#params**
ATTN-SUM-GAN$_{\text{small}}$	2	256	4	512	4.480M
SUM-GAN-AAE$_{\text{small}}$	2	256	-	-	6.510M
ATTN-SUM-GAN$_{\text{base}}$	2	512	4	1024	17.348M
SUM-GAN-AAE$_{\text{base}}$	2	512	-	-	25.472M
ATTN-SUM-GAN$_{\text{large}}$	4	512	4	1024	34.170M
SUM-GAN-AAE$_{\text{large}}$	4	512	-	-	51.203M

D. Hardware Requirements and Training Time

The hardware configuration is a single NVIDIA GeForce RTX 3050 4GB with 16GB of RAM and an 11th Gen Intel (R) CoreTM i7-11800H @ 2.30GHz CPU. For comparison, training times for the SUM-GAN-AAE and ATTN-SUM-GAN architectures are included in Table II.

TABLE II
TRAINING TIME OF ATTN-SUM-GAN AND SUM-GAN-AAE. THE REPORTED TIME IS IN FORMAT HH:MM:SS.

Model	**SumMe**	**TVSum**
ATTN-SUM-GAN$_{\text{small}}$	00:05:14	00:05:05
SUM-GAN-AAE$_{\text{small}}$	01:29:11	01:23:57
ATTN-SUM-GAN$_{\text{base}}$	00:10:50	00:10:03
SUM-GAN-AAE$_{\text{base}}$	01:29:11	02:18:04
ATTN-SUM-GAN$_{\text{large}}$	00:05:14	00:05:05
SUM-GAN-AAE$_{\text{large}}$	01:29:11	01:23:57

From Table II, it is noticeable that the training time of ATTN-SUM-GAN is 18 times faster than the SUM-GAN-AAE method, which is basically implemented on LSTM cells. With the base configuration, ATTN-SUM-GAN training time is less than SUM-GAN-AAE for 13 times. The training time is greatly reduced using the Attentive Block built on top of the Transformer encoder architecture [13].

E. Performance Benchmarking

The evaluation results are presented in Table III, indicating F-Score percentages for ATTN-SUM-GAN$_{\text{small}}$ and SUM-GAN-AAE$_{\text{small}}$ across five splits (indexing from 0 to 4), as well as the average performance across five folds.

TABLE III
F1 SCORES (%) OF THE BASELINE AND PROPOSED METHOD ON TVSUM ACROSS FIVE SPLITS AND THE AVERAGE PERFORMANCE. HIGHEST PERFORMANCE ACROSS EACH FOLDS OR IN AVERAGE IS SHOWN IN **BOLD**.

Model	**Fold**					**Avg.**
	0	**1**	**2**	**3**	**4**	
SUM-GAN-AAE$_{\text{small}}$ [16]	61.8	64.2	63.0	61.3	64.2	62.9
SUM-GAN-AAE$_{\text{base}}$ [16]	62.2	61.4	66.9	58.9	68.5	63.6
ATTN-SUM-GAN$_{\text{small}}$	62.1	61.9	65.4	63.8	64.8	63.6
ATTN-SUM-GAN$_{\text{base}}$	61.6	**65.6**	64.6	63.3	68.0	64.6
ATTN-SUM-GAN$_{\text{large}}$	**63.0**	60.7	**68.0**	**66.5**	**68.9**	**65.4**

For ATTN-SUM-GAN$_{\text{small}}$, the F-Score values range from 61.9% to 65.4%, with an overall average of 63.6%. Meanwhile, SUM-GAN-AAE$_{\text{small}}$ [16] demonstrates F-Scores ranging from 61.3% to 64.2%, resulting in an average of 62.9%. ATTN-SUM-GAN outperforms SUM-GAN-AAE [16] on four splits: 0, 2, 3, and 4, with an improvement of 0.3%, 2.4%, 2.5%, and 0.6%. However, the average performance of the ATTN-SUM-GAN$_{\text{small}}$ is equal to the one of SUM-GAN-AAE$_{\text{base}}$ [16], but the training time is less (see Table II) and the training parameters are also fewer (see Table I). Hence, it proves the improvement in term of capturing long-range dependency, as well as the decrease of training time.

When increasing the number of parameters from small setting to base setting, the proposed architecture receives a small gain in performance, increasing by 1.0% average performance from 63.6% (small configuration) to 64.6% (base configuration). The ATTN-SUM-GAN$_{\text{base}}$ outperforms ATTN-SUM-GAN$_{\text{small}}$ on only two folds: 1 and 4, but the increasing distances are large, 3.7% and 3.2% respectively, leading to a slight improvement in the base configuration compared to the small one.

Regarding the large configuration of ATTN-SUM-GAN, which has the highest number of training parameters among three proposed configuration, all of its models in five folds outperforms other configurations and the baseline, except for fold 1. The average performance of the base configuration on the TVSum dataset [15] is 65.4%.

In conclusion, the proposed methods outperforms the baseline models across three settings on the TVSum dataset [15].

F. Comparison with other methods

Besides comparing the proposed method with the baseline models [16], Table IV reports the F1 scores of other unsupervised video summarization methods compared to the proposed ATTN-SUM-GAN architecture on the TVSum dataset.

From Table IV, the proposed models outperforms all the previous methods with all of the three configurations: small, base, and large.

TABLE IV

COMPARISON OF THE PROPOSED METHOD WITH PREVIOUS UNSUPERVISED LEARNING VIDEO SUMMARIZATION WORK ON TVSUM DATASET [15]. RL DENOTES REINFORCEMENT LEARNING AND GAN DENOTES GENERATIVE ADVERSARIAL NETWORKS.

Model	**Method**	**F1 Score (%)**
SUM-GAN$_{\text{ddp}}$ [24]	GAN + LSTM	51.7
SUM-GAN$_{\text{rep}}$ [24]	GAN + LSTM	51.9
DSN [8]	RL + LSTM	57.6
CSNet [28]	GAN + Chunk and Stride	58.8
Cycle-SUM [27]	Cycle GAN + LSTM	57.6
SUM-GAN-AAE [16]	GAN + Attention	58.3
AC-SUM-GAN [31]	GAN + Actor-Critic	60.6
SUM-GDA$_{\text{unsupervised}}$ [46]	GAN + Attention	59.6
SUM-GAN-GEA [33]	GAN + Attention	61.3
ShotEncDecRL [7]	RL + Encoder-Decoder	62.2
SUM-GAN-AAE$_{\text{small}}$ (our) [16]	GAN + Attention	62.9
SUM-GAN-AAE$_{\text{base}}$ (our) [16]	GAN + Attention	63.6
ATTN-SUM-GAN$_{\text{small}}$	GAN + Transformer	63.6
ATTN-SUM-GAN$_{\text{base}}$	GAN + Transformer	64.6
ATTN-SUM-GAN$_{\text{large}}$	**GAN + Transformer**	**65.4**

V. ABLATION STUDY

This section introduces the ablation study about the effects of the different configurations of the Transformer block [13], including the hidden size, the number of heads, and the number of layers. By changing the mentioned hyperparameters, several model's performances are observed on the TVSum dataset [15] to draw a conclusion based on the empirical results. The settings of the models are taken from the small configuration (see Table I). When changing an attribute, the others are fixed.

A. Adjusting hidden size

The hidden size, also known as the number features for each keyframe, are selected from the set of $S_{\text{d}} = \{32, 64, 128, 256, 512\}$. The evaluation F1 scores of several hidden size-changed models are provided in Figure 4.

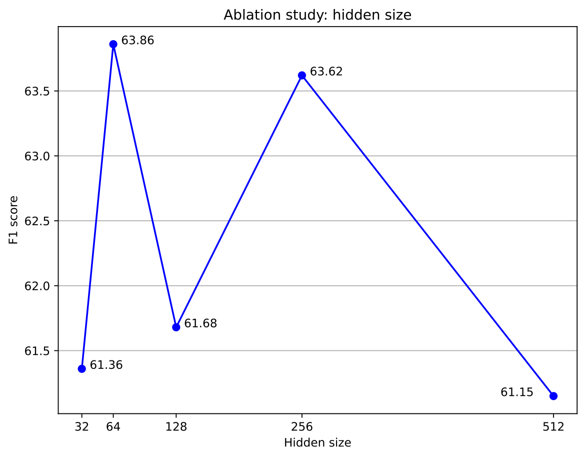


Fig. 4. Performance of models with hidden size of $\{32, 64, 128, 256, 512\}$.

In general, it is observed that the F1 scores vary across different hidden layer sizes, indicating that the choice of this parameter has an impact on the model's performance. From Figure 4, the model achieves the highest performance at 63.9% when the hidden size is 64. When increasing the hidden size to a higher value, at 128 and 512, the performance drops significantly to 61.7% and 61.2%, respectively. Only at 256, the model's performance is 63.6%, nearly equal to the one's at 64. It is noticeable that increasing the hidden size of the ATTN-SUM-GAN model does not result in increasing the performance, especially on the TVSum dataset [15]. Since larger hidden layer sizes generally require more computational resources, choosing a smaller hidden layer size (like 64) might be preferable for efficiency while preserving nearly similar performance.

In short, the empirical results indicate that 64 is the most suitable for the hidden size of the Transformer block [13] in the ATTN-SUM-GAN model. In addition, increasing the hidden size of the proposed architecture does not contribute to the model's performance on the TVSum dataset [15].

B. Changing the number of heads

To study the effect of the number of heads inside the Transformer block in the video summarization modules, the number of heads is selected with the power of two, including 2, 4, 8, 16, and 32. The performance of different model settings on the TVSum benchmark is shown in Figure 5.

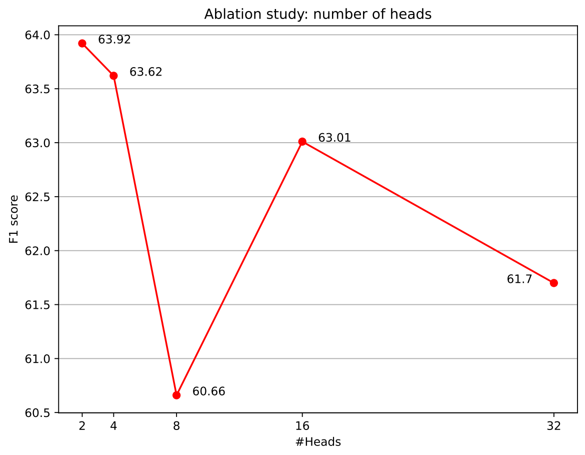


Fig. 5. Model performance with different number of heads (denoted as #Heads).

From Figure 5, it is clear that the model gets the highest performance with #Heads = 2, at 63.9%. When the number of heads increases, the model performance on TVSum [15] decreases. When the number of heads reaches 8, the model performance dramatically declines to 60.7% from 63.9% at #Heads = 4. When the number of heads is 16 or 32, the performance is still lower with respect to the setting of #Heads = 2, at 63.0% and 61.7%, respectively.

In general, the model performance observes a decreasing trend when the number of heads increases. However, there is an increase when the number of heads changes from 8 to 16.

The highest F1 score is achieved when the number of heads is 2.

VI. Conclusion

In this work, Attentive modules, including Attentive Selector, Attentive Encoder, Attentive Decoder, and Attentive Discriminator, are introduced to replace the recurrent neural networks in the generative adversarial network pipeline for unsupervised video summarization. The proposed ATTN-SUM-GAN architecture outperforms state-of-the-art methods on the TVSum dataset. Furthermore, the training time is reduced up to 18 times compared to the one where networks are built on recurrent neural networks (in particular, the SUM-GAN-AAE model). In addition, the ablation study about how the number of heads and the hidden size affect the model's performance is also examined. The empirical results indicate that a hidden size of 64 and 2 heads is the best ATTN-SUM-GAN model setting.

In the future, this work should extend to study about the integration of local and global attention into the Transformer block, changing the other kinds of positional embeddings, and evaluating the proposed ATTN-SUM-GAN on different video summarization datasets

References

[1] K. K. Kapoor, K. Tamilmani, N. P. Rana, P. Patil, Y. K. Dwivedi, and S. Nerur, "Advances in social media research: Past, present and future," *Information Systems Frontiers*, vol. 20, pp. 531–558, 2018.

[2] V. Tiwari and C. Bhatnagar, "A survey of recent work on video summarization: approaches and techniques," *Multimedia Tools and Applications*, vol. 80, no. 18, pp. 27 187–27 221, 2021.

[3] S.-S. Zang, H. Yu, Y. Song, and R. Zeng, "Unsupervised video summarization using deep non-local video summarization networks," *Neurocomputing*, vol. 519, pp. 26–35, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231222014084

[4] X. He, Y. Hua, T. Song, Z. Zhang, Z. Xue, R. Ma, N. Robertson, and H. Guan, "Unsupervised video summarization with attentive conditional generative adversarial networks," in *Proceedings of the 27th ACM International Conference on Multimedia*, ser. MM '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 2296–2304. [Online]. Available: https://doi.org/10.1145/3343031.3351056

[5] S. Jadon and M. Jasim, "Unsupervised video summarization framework using keyframe extraction and video skimming," in *2020 IEEE 5th International Conference on computing communication and automation (ICCCA)*. IEEE, 2020, pp. 140–145.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[7] Y. Yuan and J. Zhang, "Unsupervised video summarization via deep reinforcement learning with shot-level semantics," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 33, no. 1, pp. 445–456, 2022.

[8] K. Zhou, Y. Qiao, and T. Xiang, "Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018.

[9] J. Gao, X. Yang, Y. Zhang, and C. Xu, "Unsupervised video summarization via relation-aware assignment learning," *IEEE Transactions on Multimedia*, vol. 23, pp. 3203–3214, 2021.

[10] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning internal representations by error propagation," 1985.

[11] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[14] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool, "Creating summaries from user videos," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13*. Springer, 2014, pp. 505–520.

[15] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes, "Tvsum: Summarizing web videos using titles," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5179–5187.

[16] E. Apostolidis, E. Adamantidou, A. I. Metsai, V. Mezaris, and I. Patras, "Unsupervised video summarization via attention-driven adversarial learning," in *MultiMedia Modeling: 26th International Conference, MMM 2020, Daejeon, South Korea, January 5–8, 2020, Proceedings, Part I 26*. Springer, 2020, pp. 492–504.

[17] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[18] M. Basavarajaiah and P. Sharma, "Survey of compressed domain video summarization techniques," *ACM Comput. Surv.*, vol. 52, no. 6, oct 2019. [Online]. Available: https://doi.org/10.1145/3355398

[19] A. S. Parihar, R. Mittal, P. Jain, and Himanshu, "Survey and comparison of video summarization techniques," in *2021 5th International Conference on Computer, Communication and Signal Processing (ICCCSP)*, 2021, pp. 268–272.

[20] C.-W. Ngo, Y.-F. Ma, and H.-J. Zhang, "Automatic video summarization by graph modeling," in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, pp. 104–109 vol.1.

[21] A. Khosla, R. Hamid, C.-J. Lin, and N. Sundaresan, "Large-scale video summarization using web-image priors," in *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2698–2705.

[22] B. Zhao and E. P. Xing, "Quasi real-time summarization for consumer videos," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2513–2520.

[23] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, "Video summarization with long short-term memory," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*. Springer, 2016, pp. 766–782.

[24] B. Mahasseni, M. Lam, and S. Todorovic, "Unsupervised video summarization with adversarial lstm networks," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2982–2991.

[25] J. Basak, V. Luthra, and S. Chaudhury, "Video summarization with supervised learning," in *2008 19th International Conference on Pattern Recognition*. IEEE, 2008, pp. 1–4.

[26] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, "Video summarization with long short-term memory," in *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VII 14*. Springer, 2016, pp. 766–782.

[27] L. Yuan, F. E. Tay, P. Li, L. Zhou, and J. Feng, "Cycle-sum: Cycle-consistent adversarial lstm networks for unsupervised video summarization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, 2019, pp. 9143–9150.

[28] Y. Jung, D. Cho, D. Kim, S. Woo, and I. S. Kweon, "Discriminative feature learning for unsupervised video summarization," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33018537

[29] G. Liang, Y. Lv, S. Li, S. Zhang, and Y. Zhang, "Unsupervised video summarization with a convolutional attentive adversarial network," *arXiv preprint arXiv:2105.11131*, 2021.

[30] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[31] E. Apostolidis, E. Adamantidou, A. I. Metsai, V. Mezaris, and I. Patras, "Ac-sum-gan: Connecting actor-critic and generative adversarial networks for unsupervised video summarization," *IEEE Transactions on*

Circuits and Systems for Video Technology, vol. 31, no. 8, pp. 3278–3292, 2020.

[32] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, 2012.

[33] Q. Yu, H. Yu, Y. Wang, and T. D. Pham, "Sum-gan-gea: Video summarization using gan with gaussian distribution and external attention," *Electronics*, vol. 11, no. 21, p. 3523, 2022.

[34] Z. Li and L. Yang, "Weakly supervised deep reinforcement learning for video summarization with semantically meaningful reward," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2021, pp. 3239–3247.

[35] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," *arXiv preprint arXiv:1511.08458*, 2015.

[36] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (indrnn): Building a longer and deeper rnn," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5457–5466.

[37] G. Yaliniz and N. Ikizler-Cinbis, "Using independently recurrent networks for reinforcement learning based unsupervised video summarization," *Multimedia Tools and Applications*, vol. 80, pp. 17 827–17 847, 2021.

[38] T. Liu, Q. Meng, J.-J. Huang, A. Vlontzos, D. Rueckert, and B. Kainz, "Video summarization through reinforcement learning with a 3d spatio-temporal u-net," *IEEE Transactions on Image Processing*, vol. 31, pp. 1573–1586, 2022.

[39] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, 2015, pp. 234–241.

[40] L. Lebron Casas and E. Koblents, "Video summarization with lstm and deep attention models," in *International conference on multimedia modeling*. Springer, 2018, pp. 67–79.

[41] X. Feng, L. Wang, and Y. Zhu, "Video summarization with self-attention based encoder-decoder framework," in *2020 International Conference on Culture-oriented Science & Technology (ICCST)*. IEEE, 2020, pp. 208–214.

[42] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, L. Màrquez, C. Callison-Burch, and J. Su, Eds. Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 1412–1421. [Online]. Available: https://aclanthology.org/D15-1166

[43] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[46] P. Li, Q. Ye, L. Zhang, L. Yuan, X. Xu, and L. Shao, "Exploring global diverse attention via pairwise temporal relation for video summarization," *Pattern Recognition*, vol. 111, p. 107677, 2021.