Towards Automated Movie Trailer Generation

Dawit Mureja Argaw[1,2] Mattia Soldan[1] Alejandro Pardo[1] Chen Zhao[1*]
Fabian Caba Heilbron[3] Joon Son Chung[2] Bernard Ghanem[1]

[1] King Abdullah University of Science and Technology (KAUST) [2] KAIST [3] Adobe Research

Abstract

Movie trailers are an essential tool for promoting films and attracting audiences. However, the process of creating trailers can be time-consuming and expensive. To streamline this process, we propose an automatic trailer generation framework that generates plausible trailers from a full movie by automating shot selection and composition. Our approach draws inspiration from machine translation techniques and models the movies and trailers as sequences of shots, thus formulating the trailer generation problem as a sequence-to-sequence task. We introduce Trailer Generation Transformer (TGT), a deep-learning framework utilizing an encoder-decoder architecture. TGT movie encoder is tasked with contextualizing each movie shot representation via self-attention, while the autoregressive trailer decoder predicts the feature representation of the next trailer shot, accounting for the relevance of shots' temporal order in trailers. Our TGT significantly outperforms previous methods on a comprehensive suite of metrics.

1. Introduction

Movie trailers are essential marketing tools for the film industry, generating anticipation by showcasing captivating scenes, storylines, and cast members. They enable studios to fuel marketing campaigns and build interest ahead of a movie's release. For audiences, trailers serve as decision-making tools, offering a condensed preview of the film's content. Despite their importance, creating trailers is costly, time-consuming, and demands expert knowledge.

The process of creating a trailer can be broadly divided into two stages[1]. In the *first* stage, video editors immerse themselves in the entire movie, viewing all the shots. They carefully select relevant trailer shots and arrange them in a specific order to craft an engaging flow and rhythm for the movie trailer (see Fig. 1, top row). This is a time-consuming and tedious process as the editor has to sort through an extensive collection of movie shots [38]. The *second* stage involves fine-editing which incorporates dialogue and sound modeling into the trailer. This work delves into the domain of Automatic Trailer Generation (ATG) with the aim of streamlining the first stage, *i.e.* shot selection and sequencing, to create a trailer montage from a given full movie (see Fig. 1, bottom row).

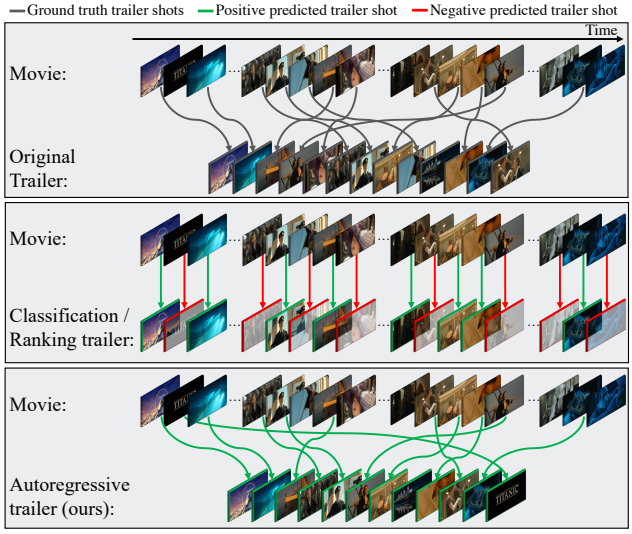


Figure 1. **Trailer Generation Problem and Solutions.** The top row depicts the movie and expert-created trailer. The process composes shots from the movie in non-chronological order to create a compelling and intriguing story. Depicted in the central row are the classification/ranking strategies that classify/rank each shot in the movie independently (classification) or with limited relative interaction (ranking). The bottom row represents our approach which can reason over the entire input movie sequence before producing a provisional trailer with non-chronological shots order.

ATG has remained a relatively underexplored problem mainly due to the complexity of the task and the lack of well-established benchmarks. Nevertheless, few prior works [6, 23, 32, 38] have made efforts to tackle this challenging task. These works have approached trailer generation in different ways. Some works [23, 32] have framed it as a binary *classification* problem, where each shot in a

*Corresponding author.
[1] https://www.youtube.com/watch?v=zkEXtwCL684

movie is classified as either a trailer shot or not, while others [6, 38] have treated it as a *ranking* problem, wherein the top-ranked movie shots are considered as trailer shots.

While the progress has been promising, previous works on trailer generation exhibit several limitations, mainly due to their problem formulations. First, both classification and ranking are likely vulnerable to a long-tail distribution problem since only a few percent of movie shots are trailer shots, and hence, there will always be a class imbalance in classification [32] or a significant amount of hard negatives in ranking [38]. Second, in classification-based works [23, 32], all trailer shots are classified in parallel, and the decision on whether a given shot is a trailer is not conditioned by shots that are already selected as trailer shots. This eventually leads to numerous repetitive shots being categorized as trailer shots. Similarly, ranking-based works [6, 38] only consider two shots at a time. Third, and most importantly, existing works do not consider shot composition (ordering), as their formulation does not allow it. Consequently, the shot sequence in the generated trailer mirrors the chronological order of the shots in the original movie. See Fig. 1, central row, for a graphical illustration of the predicted trailer generated by classification and ranking methods.

Our work addresses these limitations by introducing a new trailer generation framework. First, we formulate trailer generation as a regression problem where we predict continuous feature representations of trailer shots instead of a discrete binary class. This makes our approach less vulnerable to a long-tail distribution problem [43]. Second, we employ an autoregressive approach in which the prediction of a trailer shot embedding at a specific time step is conditioned not only on the input movie sequence but also on the generated trailer sequence up to the current step. Third, we design a model that takes shot composition into account, and we reinforce this concept through training on a large set of movie-trailer pairs with carefully designed losses.

In pursuit of these specifications, we introduce Trailer Generation Transformer (TGT), a model designed to automatically generate plausible trailers using full movies as input. Our approach uses an encoder-decoder architecture that models ATG as a sequence-to-sequence (seq2seq) problem [36, 37], where the input sequence corresponds to the movie sequence, and the target sequence represents the desired trailer. To make this process computationally feasible and follow the natural structure of movies, we utilize a shot boundary detector [35] to *tokenize* movies and trailers and use a state-of-the-art visual encoder (*i.e.* CLIP [28]) to obtain sequences of visual embeddings.

TGT's encoder comprises two components: *trailerness encoder* and *context encoder*. The *trailerness encoder* ingests the entire movie sequence and estimates the likelihood of each shot being relevant for trailer creation. This step identifies the visually captivating shots often used for trailer generation. The scores are fused with the input sequence, a process we term *trailerness encoding* drawing a parallel to positional encoding in [37]. The *context encoder* consists of a stack of transformer encoder layers, leveraging self-attention mechanisms [37] to contextualize each shot representation based on the entire movie sequence, effectively achieving temporal modeling across multiple shots.

The decoder of TGT is an autoregressive model that predicts the feature representation of the next trailer shot. This design choice is instrumental in enabling the model to learn shot composition, specifically, the ordering of shots, differentiating our approach from previous works [6, 23, 32, 38]. See the comparison between the central and bottom rows in Fig. 1. At training time, the decoder takes the encoder output and the target trailer sequence and employs a causal mask to learn to predict the next shot representation. At test time, the model generates outputs in an autoregressive manner. Finally, a greedy algorithm based on nearest neighbor retrieval matches the generated trailer shot representations with all shots in the movie, thereby selecting the most suitable shots for assembling the output trailer.

We curate an extensive dataset of movie-trailer pairs and employ tailored loss functions to train TGT for automatic trailer generation effectively. Furthermore, to foster further research on ATG, our work presents new benchmarks based on two widely-used movie datasets: MAD [33] and MovieNet [10]. We construct these benchmarks by pairing corresponding trailers and segmenting both movies and trailers into shot sequences, resulting in two datasets with movie-trailer pairs. We perform extensive experiments to assess the performance of our approach and compare it to previous methods using metrics that redefine how we measure trailer generation across various aspects.

Our contributions are: **(1)** A novel trailer generation formulation that effectively overcomes the inherent limitations of previous approaches. **(2)** Trailer Generation Transformer (TGT), an autoregressive encoder-decoder architecture that generates provisional trailers from full movies, which can be refined further by experts. **(3)** New ATG benchmarks built atop two movie datasets: MAD [33] and MovieNet [10]. We evaluate trailer generation performance using a comprehensive suite of metrics that consider all relevant aspects of the process.

2. Related Works

Trailer Generation Several papers have attempted to automate the process of trailer generation. Chen *et al.* [4] analyze movie composition based on sets of rules and grammar to generate a trailer by concatenating shots with a specific "movie tempo" above a threshold. Smeaton *et al.* [31] employ audiovisual features and a support vector machine to automatically select shots for trailers, highlighting the importance of shot sequence but leaving it to artists. Some

works [6, 9, 12] utilize text features from TV show descriptions, subtitles, or metadata, to find similar moments in the video transcript or subtitles and select corresponding shots for trailer creation. Others [14, 17, 32] use audio-visual features to identify potential trailer moments from long videos.

Some works [11, 22] tackle the trailer generation task using predefined templates and automatically fill them with clips from the initial sequence. In contrast, our approach solely relies on the movie sequence and does not use templates, additional metadata, or audio features to generate the trailer. Finally, Wang *et al.* [38] recently proposed CCANet, a deep learning model that utilizes co-attention and contrastive attention modules to match and distinguish trailer moments from non-trailer moments. However, CCANet is limited to a single genre and requires genre-specific training. Our work, on the other hand, is genre-agnostic and operates effectively across the diverse range of genres present in MovieNet [10] and MAD [33].

Video Summarization Video summarization aims to select the most important clips from a given video to create a concise video summary. Similar to trailer generation, this task involves a shot selection aspect that models must address. Earlier works attempted video summarization across different video domains without explicit supervision [2, 13, 20, 21, 25, 27, 44]. Other works approached this task using supervised learning, coming from web video summaries [8] or TV series summaries [34]. Representative works employed various techniques, including fully convolutional sequence networks that learn from paired data [30] and unpaired data [29], graph modeling [26], and determinantal point process (DPP) for structured predictions of video sequences [16, 40, 41].

Recent attempts include using attention-based encoding to score the importance of each frame using regression [5] or modeling the interaction between video and text to compute a saliency score [39]. Additionally, Zhang *et al.* [42] proposed a sequence-to-sequence modeling approach using an encoder-decoder LSTM-based architecture to preserve the video semantics in the output sequence. Gan *et al.* [7] propose leveraging movie trailers as supervision for video summarization, while our work uses pairs of movies and trailers for trailer generation rather than summarization. Lastly, CLIP-It [23] guides video summarization with text through dense video captions, while TL:DW? [24] focuses on exploring cross-modal saliency between video and text (transcript) signals to summarize instructional videos.

3. Methodology

Problem Formulation Given a movie sequence $\mathcal{M}$, the problem of automatic trailer generation (ATG) aims at generating its corresponding trailer sequence $\mathcal{T}$. Here, $\mathcal{M}$ denotes a sequence of movie shots $\{U_1, U_2, \ldots, U_n\}$ and $\mathcal{T}$ denotes a sequence of trailer shots $\{V_1, V_2, \ldots, V_m\}$, where n and m represent the number of shots in $\mathcal{M}$ and $\mathcal{T}$, respectively. During training, we use paired movie and trailer sequences, denoted as $\{\mathcal{M}_i, \mathcal{T}_i\}$, where i refers to the index of the movie-trailer pair. These pairs serve as a basis for the model to learn the essential features and patterns necessary for generating trailers from movie sequences. In contrast, during testing, the model is given only the movie sequences as input, without any corresponding trailer sequences. The purpose of this setup is to evaluate the model's ability to generate trailer sequences $\mathcal{T}'$ that are coherent and relevant to the given movie sequences $\mathcal{M}$. The performance of the model is assessed by comparing the generated trailer sequences $\mathcal{T}'$ to the ground truth trailer sequences $\mathcal{T}$, using suitable evaluation metrics.

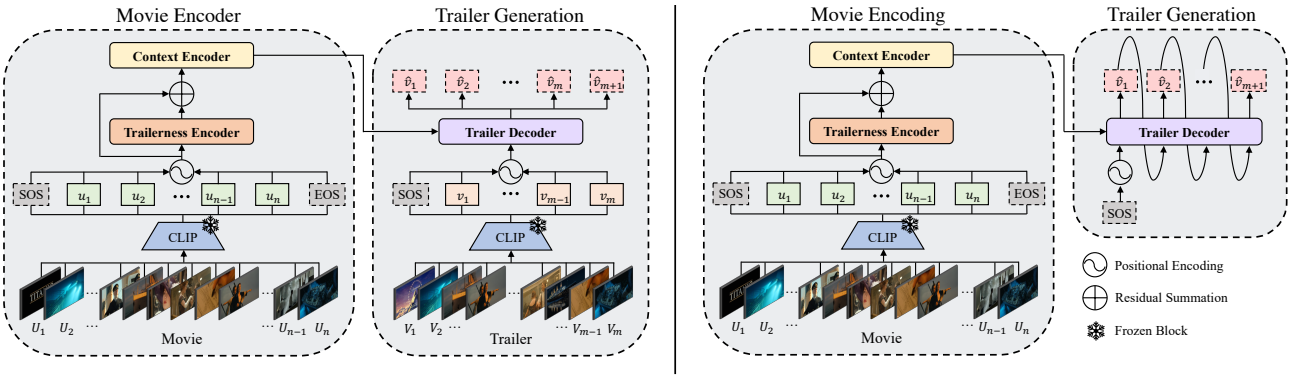
3.1. Proposed Trailer Generation Transformer

Camera shots in movies and trailers provide a convenient video structure that we leverage. Our initial step involves segmenting the movie and trailer into shot sequences ($\{U_i\}_{i=1}^n$ and $\{V_j\}_{j=1}^m$, respectively) using a state-of-the-art shot-boundary detector [35]. Learning directly from the pixel space of long-form videos in an end-to-end manner poses demanding computational burdens. Therefore, to efficiently represent the input sequences, we leverage a pretrained CLIP model [28] as a base encoder to extract the feature representations of each shot, thereby transforming the movie and trailer into a sequence of *visual tokens*, *i.e.* $\{u_i\}_{i=1}^n$ and $\{v_j\}_{j=1}^m$. To indicate the beginning and end of a movie/trailer sequence, we use learnable start-of-sequence (`SOS`) and end-of-sequence (`EOS`) tokens, respectively, as shown in Fig. 2.

We draw inspiration from machine translation techniques [3, 36, 37]. Specifically, we approach the trailer generation problem as a sequence-to-sequence (seq2seq) task [36, 37]. In seq2seq, an encoder captures semantic and syntactic information from an input token sequence and converts it into a contextualized sequence. Then, a decoder generates the translated sequence by producing tokens one at a time, conditioned on the contextualized sequence and previously generated words. We apply this concept to model the automatic trailer generation (ATG) problem, essentially "translating" a movie into a trailer. To do this, we use an attention-based encoder-decoder architecture [37] and introduce our Trailer Generation Transformer (TGT) model. TGT consists of two main components: the Movie Encoder (detailed in Sec. 3.1.1) and an auto-regressive Trailer Decoder (described in Sec. 3.1.2).

3.1.1 Movie Encoder

The movie encoder aims at encoding a given movie sequence into a contextualized feature sequence by using a trailerness encoder followed by a context encoder, as shown



(a) TGT training pipeline.

(b) TGT inference pipeline.

Figure 2. **Architecture Overview.** Subfigure (a) illustrates our TGT model's training pipeline. Movies are segmented into shots and transformed into visual embeddings via a pre-trained CLIP model [28]. Enhanced with positional embeddings and trailerness scores, these tokens undergo context encoding. The trailer decoder, during training, uses ground-truth trailer shots as queries for cross-attention with encoder output, then parallelly regresses the next shot feature using a causal mask. Subfigure (b) shows the inference pipeline where the trailer decoder sequentially generates trailer shots in an autoregressive manner while the movie encoder process remains unchanged.

in the Movie Encoder block of Fig. 2.

Trailerness Encoder Given an input sequence of visual tokens representing a movie, $\{\texttt{SOS}, u_1, \ldots, u_n, \texttt{EOS}\}$, a positional encoding layer is first used to embed information about the relative position of the input tokens. We then feed the sequence into a trailerness encoder $\mathcal{E}_\mathcal{T}$. The purpose of the trailerness encoder is to observe the full movie and reason over the likeliness of each movie shot to be in a trailer. In $\mathcal{E}_\mathcal{T}$, a Transformer [37] self-attention layer $\mathcal{A}$, followed by a linear layer f and a Sigmoid layer s, is used to predict a trailerness score for each visual token in the input sequence, as formulated below:

$$\{t_i^{\text{p}}\}_{i=0}^{n+1} = s(f(\mathcal{A}(\texttt{SOS}+\sigma_0, u_1+\sigma_1, \ldots, \texttt{EOS}+\sigma_{n+1}))), \quad (1)$$

where σ_i denotes the positional encoding value at position i and $t_i^{\text{p}} \in [0, 1]$ represents the predicted score of a movie shot U_i as a noteworthy shot to be included in a trailer. During training, we compute the ground truth trailerness score t_i^{GT} for each movie shot by taking the maximum value of the cosine similarity between the movie shot and all the trailer shots (see Eq. (2)). A high value of t_i^{GT} means that a movie shot U_i is in a trailer, whereas a low value indicates that the movie shot is not similar to any shots in the trailer. We optimize the mean-squared error between the predicted and ground truth trailerness scores during training to guide the trailerness encoder $\mathcal{E}_\mathcal{T}$ as shown below in Eq. (3)

$$t_i^{\text{GT}} = \max_j \; s_{i,j}, \quad \text{for } 1 \leq i \leq n, \text{where } s_{i,j} = \frac{u_i \cdot v_j}{\|u_i\| \|v_j\|}, \quad (2)$$

$$\mathcal{L}_t = \sum_{i=0}^{n+1} |t_i^{\text{p}} - t_i^{\text{GT}}|^2, \quad (3)$$

where $\mathcal{L}_t$ denotes trailerness loss and the ground truth trailerness score for $\texttt{SOS}$ (t_0^{GT}) and $\texttt{EOS}$ (t_{n+1}^{GT}) tokens is set to 0. After positional encoding, the predicted trailerness scores are added to inject information about the relative degree of trailerness of the movie shots in the sequence (see Fig. 2 Movie Encoder).

Context Encoder Trailerness encoding adjusts each positionally encoded visual token by a unique float number, representing the trailerness score. Yet, to understand the relationships between shots in the movie sequence, the trailerness encoded sequence is passed through a context encoder $\mathcal{E}_\mathcal{C}$ (see Eq. (4)). The purpose of the context encoder is to obtain a contextualized representation of the input sequence, essential for decoding a trailer. We use a stack of Transformer [37] encoder layers for $\mathcal{E}_\mathcal{C}$. Through the multi-head self-attention mechanism, $\mathcal{E}_\mathcal{C}$ considers the context in which each movie shot appears, generating a representation that encapsulates the meaning of the entire movie sequence.

$$C = \mathcal{E}_\mathcal{C}(\texttt{SOS}+\sigma_0+t_0^{\text{p}}, u_1+\sigma_1+t_1^{\text{p}}, \ldots, \texttt{EOS}+\sigma_{n+1}+t_{n+1}^{\text{p}}), \quad (4)$$

where C denotes a contextualized movie sequence representation from $\mathcal{E}_C$.

3.1.2 Trailer Decoder

Given the output of the context encoder, we utilize a trailer decoder $\mathcal{D}_\mathcal{T}$ to autoregressively generate shot embeddings, which are subsequently matched back to the movie sequence to compose the trailer. At training time, the trailer decoder inputs the learned context C and a positionally encoded but right-shifted trailer sequence as shown in Fig. 2 (a). The decoding is done in a sequential manner that $\mathcal{D}_\mathcal{T}$ attends to the context C and a trailer sequence up to the current position $\{\texttt{SOS}, v_1, \ldots, v_{j-1}\}$ in order to output the next trailer embedding $\hat{v}_j$ (Eq. (5)). This enables the network to efficiently capture sequential dependencies and positional information of shots, facilitating the learning of shot composition for generating a trailer sequence. The next trailer shot generation scheme in our work is equivalent to

next-word prediction in language modeling [37]. We use a stack of Transformer [37] decoder layers for $\mathcal{D}_\mathcal{T}$. At inference time, as ground truth trailer sequence is not available, the trailer decoder starts with $(C, \{\texttt{SOS}\})$ and autoregressively decodes a trailer sequence using the previously generated trailer shot embeddings until decoding the EOS token. The decoding process is formulated as follows:

$$\hat{v}_j = \mathcal{D}_\mathcal{T}(C, \{\texttt{SOS}, v_1, \ldots, v_{j-1}\}), \quad 1 \leq j \leq m{+}1. \quad (5)$$

Lastly, we adopt a Greedy Search strategy utilizing Nearest Neighbor retrieval. This method involves comparing each embedding of the decoded shots against all shot representations from the movie. The shot with the highest feature similarity is then selected for assembling the output trailer.

3.1.3 Training Losses

We train our network by optimizing the embedding distance between the predicted trailer sequence $\{\hat{v}_1, \ldots, \hat{v}_m, \hat{v}_{m+1}\}$ and the ground truth sequence $\{v_1, \ldots, v_m, \texttt{EOS}\}$. To ensure that the trailer decoder outputs the correct shot at each step, we minimize the reconstruction loss between each predicted trailer embedding and the corresponding ground truth embedding as shown in Eq. (6). As mentioned earlier, the order in which the shots are decoded is crucial for trailer generation. To integrate this concept into the proposed model, we introduce a sequence-based loss, minimizing the Kullback-Leibler (KL) divergence loss between the predicted trailer sequence distribution and the ground truth distribution, as shown in Eq. (7).

$$\mathcal{L}_{\text{rec}} = \sum_{j=1}^{m+1} |\hat{v}_j - v_j|^2, \quad (6)$$

$$\mathcal{L}_{\text{KL}} = \sum_j \texttt{softmax}(v_j) \cdot \log\left(\frac{\texttt{softmax}(v_j)}{\texttt{softmax}(\hat{v}_j)}\right). \quad (7)$$

The total training loss is defined as the sum of the trailerness encoding loss in Eq. (3), the feature reconstruction loss in Eq. (6), and the sequence-based loss Eq. (7).

$$\mathcal{L}_{\text{total}} = \mathcal{L}_t + \mathcal{L}_{\text{rec}} + \mathcal{L}_{\text{KL}}. \quad (8)$$

3.2. Datasets

Movies exhibit a large diversity encompassing genre, style, narrative structure, theme, artistic choices, and many more. We argue that exposing our model to a large and comprehensive collection of movies is crucial for building a generalizable trailer generation pipeline. Following this motivation, we collect a training set of 23,304 movies and trailers spanning 28 genres. These movies cover over 120 years of cinematic history, albeit with a natural bias towards more recent films, reflecting the industry's continuous growth and increased production rates. The training set is paired with a validation split containing 300 movie-trailer pairs used for hyperparameter selection.

Furthermore, we introduce new benchmarks for trailer generation, building upon two mainstream movie datasets: MAD [33] and MovieNet [10]. To align these datasets with our task, we enhance them by including their respective trailers obtained from IMDb [1]. Notably, not all movies in these datasets have trailers available on their IMDb pages. Consequently, we obtained 602 out of 650 movie-trailer pairs for MAD and 989 out of 1100 for MovieNet. We also ensure no overlap between the test sets and the training/validation splits.

3.3. Evaluation Metrics

We evaluate the trailer generation performance by taking both shot selection and sequencing into account. We follow previous works [9, 38] and employ Precision, Recall, and F1-score metrics to measure the accuracy of a model in predicting trailer shots from an input movie sequence. We also measure the correctness of the predicted trailer sequence using Levenshtein distance (LD) and Sequence length difference (SLD) metrics. Levenshtein distance (also known a edit distance) [15] is defined as the minimum number of edits (insertions, deletions, or substitutions) required to transform the decoded trailer sequence into the ground truth sequence. Sequence length difference measures the absolute difference in the number of shots between the predicted trailer sequence and the ground truth sequence.

4. Experiment

Implementation Details We use the TransNet-V2 [35] model for preprocessing movies and trailers into sequences of shots. Additionally, we employ the pretrained CLIP ViT H/14 [28] for shot feature extraction. The trailerness encoder $\mathcal{E}_\mathcal{T}$ contains a single Transformer [37] encoder layer. The context encoder $\mathcal{E}_\mathcal{C}$ consists of four encoder layers. Five decoder layers are used for the trailer decoder $\mathcal{D}_\mathcal{T}$. Each encoder and decoder layer has a hidden dimension of 1024, 8 attention heads, and a feed-forward dimension of 2048. SOS and EOS tokens are initialized by creating an embedding vector of size 1024, sampled from a Normal distribution. AdamW [19] optimizer with an initial learning rate of $1e-4$ and a cosine warm-up scheduler is used during training. Our model is trained for 200 epochs with a mini-batch size of 8, on a single NVIDIA A100 GPU [2].

4.1. Trailer Generation Performance

In this section, we evaluate the performance of our approach by comparing it with three baseline methods: Random, CCANet [38] and CLIP-It [23], on the MAD [33] and

[2] Training experiments done at KAUST.

Table 1. **Experimental comparison with different baselines.**

	MAD [33]					MovieNet [10]				
	Precision ↑	Recall ↑	F1-score ↑	LD ↓	SLD ↓	Precision ↑	Recall ↑	F1-score ↑	LD ↓	SLD ↓
Random	5.32	6.05	5.65	-	-	4.96	5.67	5.28	-	-
CCANet [38]	32.15	30.76	31.63	81.25	-	30.46	29.29	29.58	90.18	-
CLIP-It [23]	40.29	43.05	41.73	95.58	47.10	38.19	40.28	39.34	103.64	51.30
TGT (ours)	**55.30**	**49.92**	**52.38**	**21.18**	**10.78**	**49.74**	**44.32**	**46.77**	**24.66**	**13.28**

Figure 3. **Subjective quality of our trailer generation.** We compare a movie trailer against the one produced by our TGT method. We highlight in green the correctly selected shots, in orange shots that are visually similar, and in red mismatched shots.

MovieNet [10] benchmarks. To establish a Random baseline, we randomly select m shots from each movie, where m represents the total number of shots present in the corresponding ground truth trailer. As shown in Table 1, a baseline that randomly selects movie shots for trailer generation exhibits significantly poor performance across all metrics.

Comparison to Trailer Generation SoTA CCANet [38] employs a co-attention module to rank a set of movie shots in a contrastive manner, where shots with higher rankings are considered to be trailer moments. To assess the performance of CCANet for trailer generation, we select the top m ranked shots from a given movie and utilize the resulting sequence for evaluation purposes. As evident from Table 1, our approach outperforms CCANet by a large margin. For instance, on the F1-score metric, TGT achieves an average accuracy of 52.38% and 46.77% on MAD [33] and MovieNet [10] datasets, respectively. In comparison, CCANet could only achieve 31.63% and 29.58%. This is mainly because CCANet has a limited attention span, focusing on two shots at a time, which hinders accurate rankings due to the abundance of negative samples in full-length movies. In contrast, our approach generates trailers by simultaneously considering the entire movie sequence. It can also be inferred from Table 1 that the trailer sequence decoded by TGT shows much greater similarity to the ground truth sequence ($\sim$21 edits on LD metric on MAD dataset) compared to CCANet ($\sim$81 edits) even though the LD metric favors CCANet due to the assumption of an equal sequence length to the ground truth. This is an expected observation since CCANet [38] mainly targets trailer moment detection and does not take shot sequencing into account.

Comparison to Video Summarization SoTA We conduct further comparison by adapting a state-of-the-art video summarization model, CLIP-It [23], for trailer generation. CLIP-It utilizes a vanilla Transformer [37] architecture, where the input sequence is passed through both the encoder and decoder. Additionally, a binary classifier is employed to identify summary-worthy inputs in the sequence. In a parallel fashion, we classify each shot in a movie as either belonging to the trailer or not. The shots classified as trailer shots are subsequently merged to construct the final trailer. The results presented in Table 1 suggest that the CLIP-It [23] based method generally achieves better performance compared to CCANet [38]. However, it is worth highlighting that our approach notably outperforms CLIP-It. For example, on the Recall metric, TGT outperforms CLIP-It by 15.95% and 10.03% on MAD [33] and MovieNet [10] benchmarks, respectively. This is likely because CLIP-It [23] primarily focuses on shot-level classification and may not fully capture the temporal correlations between shots necessary for creating a compelling trailer. In contrast, our autoregressive approach enables TGT to attend to the contextual dependencies between shots, aligning with the movie's narrative and dramatic flow to generate a coherent and structured trailer. In terms of the predicted trailer length, it is evident from Table 1 that CLIP-It deviates by $\sim$51 shots on the MovieNet dataset, whereas TGT only deviates by $\sim$11 shots on average.

Qualitative Results In Fig. 3, we qualitatively compare a trailer decoded by our approach with its corresponding ground truth sequence. For visualization purposes, we select a subsequence from the original trailer. The shots enclosed in green boxes represent accurate predictions, while those in orange boxes indicate resembling predictions. The shots in red boxes correspond to failed predictions. The results in Fig. 3 indicate that the proposed method outputs a reasonable trailer sequence from a given movie unseen during training. For instance, TGT generates a third shot

Table 2. **Ablation experiment on network components.** $\mathcal{E}_\mathcal{T}$: trailerness encoder; $\mathcal{E}_\mathcal{C}$: context encoder. It shows both encoders contribute to TGT performance, and $\mathcal{E}_\mathcal{C}$ is especially important.

		Precision ↑	Recall ↑	F1-score ↑	LD ↓
MAD	w/o $\mathcal{E}_\mathcal{T}$	53.23	47.64	50.20	22.72
	w/o $\mathcal{E}_\mathcal{C}$	43.36	36.07	39.23	25.70
	w/o $\mathcal{E}_\mathcal{T} + \mathcal{E}_\mathcal{C}$	42.36	36.15	38.89	25.75
	w/ $\mathcal{E}_\mathcal{T} + \mathcal{E}_\mathcal{C}$ (**TGT**)	**55.30**	**49.92**	**52.38**	**21.18**
MovieNet	w/o $\mathcal{E}_\mathcal{T}$	48.38	41.80	44.75	25.62
	w/o $\mathcal{E}_\mathcal{C}$	40.08	33.41	36.32	27.09
	w/o $\mathcal{E}_\mathcal{T} + \mathcal{E}_\mathcal{C}$	39.01	33.01	35.64	27.86
	w/ $\mathcal{E}_\mathcal{T} + \mathcal{E}_\mathcal{C}$ (**TGT**)	**49.74**	**44.32**	**46.77**	**24.66**

that closely resembles the corresponding insert shot (black frame with text) in the ground truth, even though the movie sequence itself does not contain that specific shot.

4.2. Ablation Studies

Network Components In Table 2, we present ablation experiments on the different network components used in the proposed framework. First, we study the importance of trailerness encoding by training our network without $\mathcal{E}_\mathcal{T}$, *i.e.* we directly feed the positionally encoded movie sequence into the context encoder. As can be inferred from Table 2, TGT trained without $\mathcal{E}_\mathcal{T}$ still gives a competitive performance across all metrics. This is intuitive since the context encoder has to implicitly capture the trailerness of each shot as it is trained to learn a contextualized representation for the trailer decoder. However, explicitly embedding relative trailerness scores (predicted from $\mathcal{E}_\mathcal{T}$) into the movie sequence has resulted in a better performance. For instance, on the F1-score metric, trailerness encoding results in a 4.35% and 4.53% performance improvement on MAD [33] and MovieNet [10].

To investigate the benefit of learning contextualized movie representation for trailer generation, we train our network without the context encoder $\mathcal{E}_\mathcal{C}$. To do so, we use the trailerness encoded movie sequence as a context and feed it into the trailer decoder. As shown in Table 2, training TGT without $\mathcal{E}_\mathcal{C}$ results in a notably worse performance. For example, on the Recall metric, TGT without $\mathcal{E}_\mathcal{C}$ could only achieve a top-1 accuracy of 36.07% and 33.41% on MAD [33] and MovieNet [10] datasets, respectively. In comparison, TGT with $\mathcal{E}_\mathcal{C}$ gives an accuracy of 49.92% and 44.32%. It can be observed from Table 2 that a similar performance drop happens when using a decoder-only architecture, *i.e.* the positionally encoded movie sequence is fed into the trailer decoder $\mathcal{D}_\mathcal{T}$ (without $\mathcal{E}_\mathcal{T} + \mathcal{E}_\mathcal{C}$). These results demonstrate the challenge of directly generating a trailer sequence from a movie sequence and highlight the importance of obtaining a contextualized representation of the movie for effective trailer decoding.

Loss Functions Here, we examine the contribution of various loss functions used to train our network. First, we train TGT using the reconstruction loss $\mathcal{L}_\text{rec}$ in Eq. (6) as the only loss function. As can be seen from Table 3, $\mathcal{L}_\text{rec}$ is a strong enough constraint to train a competitive baseline model. We also investigate the benefit of optimizing the predicted trailerness scores with ground truth scores. It can be inferred from Table 3 that TGT trained with trailerness loss $\mathcal{L}_t$ in Eq. (3) performs consistently better compared to TGT trained without $\mathcal{L}_t$. For instance, on the Precision metric, a network trained with $\mathcal{L}_\text{rec} + \mathcal{L}_t$ outperforms a network trained with $\mathcal{L}_\text{rec}$ by 5.25% and 2.46% on MAD [33] and MovieNet [10] datasets, respectively.

Table 3. **Ablation experiment on loss functions.** $\mathcal{L}_\text{rec}$: reconstruction loss of trailer embeddings; $\mathcal{L}_t$: trailerness loss; $\mathcal{L}_\text{KL}$: KL divergence loss of trailer sequence distribution. It shows that both $\mathcal{L}_t$ and $\mathcal{L}_\text{KL}$ are beneficial, and using them together with $\mathcal{L}_\text{rec}$ leads to the highest performance.

		Precision ↑	Recall ↑	F1-score ↑	LD ↓
MAD	$\mathcal{L}_\text{rec}$	49.74	43.61	46.38	26.81
	$\mathcal{L}_\text{rec} + \mathcal{L}_t$	52.35	46.06	48.90	25.84
	$\mathcal{L}_\text{rec} + \mathcal{L}_\text{KL}$	53.86	48.63	51.03	21.99
	$\mathcal{L}_\text{rec} + \mathcal{L}_t + \mathcal{L}_\text{KL}$ (**TGT**)	**55.30**	**49.92**	**52.38**	**21.18**
MovieNet	$\mathcal{L}_\text{rec}$	45.40	39.32	42.04	31.84
	$\mathcal{L}_\text{rec} + \mathcal{L}_t$	46.52	40.15	42.99	30.91
	$\mathcal{L}_\text{rec} + \mathcal{L}_\text{KL}$	48.48	43.19	45.60	25.19
	$\mathcal{L}_\text{rec} + \mathcal{L}_t + \mathcal{L}_\text{KL}$ (**TGT**)	**49.74**	**44.32**	**46.77**	**24.66**

We investigate the impact of minimizing the KL divergence between the predicted trailer sequence and the ground truth sequence as defined in Eq. (7). The results in Table 3 show that training with $\mathcal{L}_\text{KL}$ results in a considerable performance gain. Particularly, on the Levenshtein distance (LD) metric, a trailer decoded by TGT trained with $\mathcal{L}_\text{KL}$ requires an average of 4.83 and 6.65 fewer edit steps (to change it to the ground truth sequence) on MAD [33] and MovieNet [10] datasets, respectively, compared to TGT trained without $\mathcal{L}_\text{KL}$. These results emphasize the significance of the KL-divergence loss $\mathcal{L}_\text{KL}$ in ensuring that the trailer sequence generated by TGT closely matches the ground truth sequence. As can be inferred from Table 3, the combination of all training losses ($\mathcal{L}_\text{rec} + \mathcal{L}_t + \mathcal{L}_\text{KL}$) yields the best network performance.

4.3. Experimental Analyses

Text-Controlled Trailer Generation Thus far, we have formulated movie trailer generation as a video-to-video problem. Here, we experiment with incorporating language into TGT to guide trailer decoding. For this purpose, we obtain the plot summary of each movie from IMDb [1] and encode the text using a pretrained RoBERTa [18] model. We explore two types of network variants for text-conditioned trailer generation. First, we simply concatenate the encoded text with the output of the context encoder as a control to enrich the context pool of TGT, and the resulting representation will be fed into the trailer decoder (referred to as "en-

Table 4. **Analysis on text-controlled trailer generation.** "Encoded text" leads to a significant improvement across all metrics. Using "contextualized text" further improves the performance.

		Precision ↑	Recall ↑	F1-score ↑	LD ↓
MAD	TGT	55.30	49.92	52.38	21.18
	+ Encoded text	59.27	54.68	56.81	19.34
	+ Contextualized text	**60.93**	**56.44**	**58.52**	**18.40**
MovieNet	TGT	49.74	44.32	46.77	24.66
	+ Encoded text	53.69	49.05	51.19	23.59
	+ Contextualized text	**56.13**	**51.65**	**53.72**	**21.81**

Table 5. **Analysis on trailer shot selection.** Relaxing the metric enhances performance, indicating the potential of TGT to be used in collaboration with a human editor.

		Precision ↑	Recall ↑	F1-score ↑	LD ↓
MAD	Top-1	55.30	49.92	52.38	21.18
	Top-5	65.31	60.57	62.76	15.51
	Top-10	**69.02**	**64.85**	**66.79**	**13.05**
MovieNet	Top-1	49.74	44.32	46.77	24.66
	Top-5	58.58	54.05	56.12	18.86
	Top-10	**61.84**	**57.74**	**59.63**	**16.23**

coded text" in Table 4). Second, we experiment with reasoning over the encoded text before feeding it to the trailer decoder. We use a single Transformer [37] encoder layer to obtain a contextualized text representation (referred to as "contextualized text" in Table 4). As evident from Table 4, "encoded text" leads to a significant improvement in trailer decoding performance across all metrics. Using "contextualized text" on top of pretrained RoBERTa [18] further improves trailer generation performance. This finding is consistent with our expectation that additional context enhances generation quality. More importantly, as a plot summary typically includes major plot points such as the setting, characters, and conflicts, it enables the trailer decoder to establish a more structured trailer decoding rule by attending to the relevant movie shots.

Shot Selection One of the key challenges of generating a trailer from a movie is shot selection. At each step of decoding, there could be multiple movie shot candidates that are semantically similar to the target trailer shot. In Table 5, we evaluate our model's performance by analyzing the quality of the shots decoded at each step, considering the top-5 and top-10 matching shots. As anticipated, expanding the pool of shot candidates leads to enhanced model performance across all evaluated metrics. For instance, on MAD [33] benchmark, TGT achieves a top-10 precision of 69.02% and the resulting trailer is only ∼13 edit steps away from the ground truth trailer. This is particularly noteworthy as it showcases the potential of our model to serve as a shot recommender during the editing process, where a human editor collaborates with the model to make informed decisions.

Table 6. **Analysis on the training data size.** Performance drops significantly when using only 10% of the data, while using 50% of the data yields decent performance compared to the full dataset.

		Precision ↑	Recall ↑	F1-score ↑	LD ↓
MAD	10 %	35.23	33.25	34.11	27.34
	50 %	50.30	47.71	48.80	24.32
	100 %	**55.30**	**49.92**	**52.38**	**21.18**
MovieNet	10 %	28.38	26.81	27.46	30.84
	50 %	44.71	41.61	42.96	27.43
	100 %	**49.74**	**44.32**	**46.77**	**24.66**

Scaling Effect In Table 6, we present the results of a scaling experiment conducted to evaluate the performance of our approach using different training data sizes, specifically 10%, 50%, and 100% of the available data. As can be inferred from the table, model performance directly correlates with the size of the training data. This observation is intuitive, as a larger dataset enables the model to encompass a broader range of patterns and variations found in movie and trailer sequences.

4.4. Discussion and Limitations

The proposed TGT method streamlines the trailer creation process by enabling efficient selection and ordering of shots. However, the current method does not incorporate dialogue and sound modeling, which are crucial for fine editing. This limitation could be addressed in future work by incorporating these additional factors into the TGT model. Despite this limitation, the TGT method can arguably provide significant time-saving to editors by automating the initial steps of shot selection and ordering. This shift in workload empowers editors to focus on the artistic aspects of trailer creation, such as refining cuts, durations, and injecting subtle audio elements to enhance the trailer further. Therefore, the TGT method, while limited in scope, has the potential to significantly improve the efficiency and creativity of trailer creation.

5. Conclusion

This work presents a novel approach to automatic trailer generation. Our TGT models the task as a machine translation problem and uses an effective encode-decoder architecture to generate plausible trailers. We present two newly constructed benchmarks and show that TGT outperforms state-of-the-art approaches. We believe this study has the potential to advance video summarization and promotional content creation across various domains.

Acknowledgement This work was supported by the King Abdullah University of Science and Technology (KAUST) Office of Sponsored Research through the Visual Computing Center (VCC) funding, as well as the SDAIA-KAUST Center of Excellence in Data Science and Artificial Intelligence (SDAIA-KAUST AI).

References

[1] Imdb. https://www.imdb.com/. Accessed: March 15, 2023. 5, 7

[2] Taivanbat Badamdorj, Mrigank Rochan, Yang Wang, and Li Cheng. Contrastive learning for unsupervised video highlight detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14042–14052, 2022. 3

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 3

[4] Hsuan-Wei Chen, Jin-Hau Kuo, Wei-Ta Chu, and Ja-Ling Wu. Action movies segmentation and summarization based on tempo analysis. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 251–258, 2004. 2

[5] Jiri Fajtl, Hajar Sadeghi Sokeh, Vasileios Argyriou, Dorothy Monekosso, and Paolo Remagnino. Summarizing videos with attention. In *Computer Vision–ACCV 2018 Workshops: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers 14*, pages 39–54. Springer, 2019. 3

[6] Bhagyashree Gaikwad, Ankita Sontakke, Manasi Patwardhan, Niranjan Pedanekar, and Shirish Karande. Plots to previews: Towards automatic movie preview retrieval using publicly available meta-data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 3205–3214, 2021. 1, 2, 3

[7] Bei Gan, Xiujun Shu, Ruizhi Qiao, Haoqian Wu, Keyu Chen, Hanjun Li, and Bo Ren. Collaborative noisy label cleaner: Learning scene-aware trailers for multi-modal highlight detection in movies. *arXiv preprint arXiv:2303.14768*, 2023. 3

[8] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc 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*, pages 505–520. Springer, 2014. 3

[9] Mohammad Hesham, Bishoy Hani, Nour Fouad, and Eslam Amer. Smart trailer: Automatic generation of movie trailer using only subtitles. In *2018 First International Workshop on Deep and Representation Learning (IWDRL)*, pages 26–30. IEEE, 2018. 3, 5

[10] Qingqiu Huang, Yu Xiong, Anyi Rao, Jiaze Wang, and Dahua Lin. Movienet: A holistic dataset for movie understanding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 709–727. Springer, 2020. 2, 3, 5, 6, 7

[11] Go Irie, Takashi Satou, Akira Kojima, Toshihiko Yamasaki, and Kiyoharu Aizawa. Automatic trailer generation. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 839–842, 2010. 3

[12] Yoshihiko Kawai, Hideki Sumiyoshi, and Nobuyuki Yagi. Automated production of tv program trailer using electronic program guide. In *Proceedings of the 6th ACM international conference on Image and video retrieval*, pages 49–56, 2007. 3

[13] Aditya Khosla, Raffay Hamid, Chih-Jen Lin, and Neel Sundaresan. Large-scale video summarization using web-image priors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2698–2705, 2013. 3

[14] Allyson King, Eric Zavesky, and Michael J Gonzales. User preferences for automated curation of snackable content. In *26th International Conference on Intelligent User Interfaces*, pages 270–274, 2021. 3

[15] Vladimir I Levenshtein et al. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, pages 707–710. Soviet Union, 1966. 5

[16] Yandong Li, Liqiang Wang, Tianbao Yang, and Boqing Gong. How local is the local diversity? reinforcing sequential determinantal point processes with dynamic ground sets for supervised video summarization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 151–167, 2018. 3

[17] Xingchen Liu and Jianming Jiang. Semi-supervised learning towards computerized generation of movie trailers. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2990–2995. IEEE, 2015. 3

[18] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 7, 8

[19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5

[20] Zheng Lu and Kristen Grauman. Story-driven summarization for egocentric video. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2714–2721, 2013. 3

[21] Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic. Unsupervised video summarization with adversarial lstm networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 202–211, 2017. 3

[22] Prakhar Mishra, Chaitali Diwan, Srinath Srinivasa, and G Srinivasaraghavan. A semi-automatic approach for generating video trailers for learning pathways. In *Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners' and Doctoral Consortium: 23rd International Conference, AIED 2022, Durham, UK, July 27–31, 2022, Proceedings, Part II*, pages 302–305. Springer, 2022. 3

[23] Medhini Narasimhan, Anna Rohrbach, and Trevor Darrell. Clip-it! language-guided video summarization. *Advances in Neural Information Processing Systems*, 34:13988–14000, 2021. 1, 2, 3, 5, 6

[24] Medhini Narasimhan, Arsha Nagrani, Chen Sun, Michael Rubinstein, Trevor Darrell, Anna Rohrbach, and Cordelia Schmid. Tl; dw? summarizing instructional videos with task relevance and cross-modal saliency. In *European Conference on Computer Vision*, pages 540–557. Springer, 2022. 3

[25] Rameswar Panda and Amit K Roy-Chowdhury. Collaborative summarization of topic-related videos. In *Proceedings of*

the IEEE Conference on computer vision and pattern recognition, pages 7083–7092, 2017. 3

[26] Jungin Park, Jiyoung Lee, Ig-Jae Kim, and Kwanghoon Sohn. Sumgraph: Video summarization via recursive graph modeling. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXV 16*, pages 647–663. Springer, 2020. 3

[27] Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid. Category-specific video summarization. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pages 540–555. Springer, 2014. 3

[28] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021. 2, 3, 4, 5

[29] Mrigank Rochan and Yang Wang. Video summarization by learning from unpaired data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7902–7911, 2019. 3

[30] Mrigank Rochan, Linwei Ye, and Yang Wang. Video summarization using fully convolutional sequence networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 347–363, 2018. 3

[31] Alan F Smeaton, Bart Lehane, Noel E O'Connor, Conor Brady, and Gary Craig. Automatically selecting shots for action movie trailers. In *Proceedings of the 8th ACM international workshop on Multimedia information retrieval*, pages 231–238, 2006. 2

[32] John R Smith, Dhiraj Joshi, Benoit Huet, Winston Hsu, and Jozef Cota. Harnessing ai for augmenting creativity: Application to movie trailer creation. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1799–1808, 2017. 1, 2, 3

[33] Mattia Soldan, Alejandro Pardo, Juan León Alcázar, Fabian Caba, Chen Zhao, Silvio Giancola, and Bernard Ghanem. Mad: A scalable dataset for language grounding in videos from movie audio descriptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5026–5035, 2022. 2, 3, 5, 6, 7, 8

[34] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. Tvsum: Summarizing web videos using titles. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5179–5187, 2015. 3

[35] Tomáš Souček and Jakub Lokoč. Transnet v2: an effective deep network architecture for fast shot transition detection. *arXiv preprint arXiv:2008.04838*, 2020. 2, 3, 5

[36] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014. 2, 3

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 2, 3, 4, 5, 6, 8

[38] Lezi Wang, Dong Liu, Rohit Puri, and Dimitris N Metaxas. Learning trailer moments in full-length movies with co-contrastive attention. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*, pages 300–316. Springer, 2020. 1, 2, 3, 5, 6

[39] Yifang Xu, Yunzhuo Sun, Yang Li, Yilei Shi, Xiaoxiang Zhu, and Sidan Du. Mh-detr: Video moment and highlight detection with cross-modal transformer. *arXiv preprint arXiv:2305.00355*, 2023. 3

[40] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman. Summary transfer: Exemplar-based subset selection for video summarization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1059–1067, 2016. 3

[41] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen 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*, pages 766–782. Springer, 2016. 3

[42] Ke Zhang, Kristen Grauman, and Fei Sha. Retrospective encoders for video summarization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 383–399, 2018. 3

[43] Yifan Zhang, Bingyi Kang, Bryan Hooi, Shuicheng Yan, and Jiashi Feng. Deep long-tailed learning: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 2

[44] Kaiyang Zhou, Yu Qiao, and Tao Xiang. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 3