AutoAD II: The Sequel – Who, When, and What in Movie Audio Description

Tengda Han[1] Max Bain[1] Arsha Nagrani[1†] Gül Varol[1,2] Weidi Xie[1,3] Andrew Zisserman[1]
[1]Visual Geometry Group, University of Oxford [2]LIGM, École des Ponts ParisTech [3]CMIC, Shanghai Jiao Tong University

Abstract

Audio Description (AD) is the task of generating descriptions of visual content, at suitable time intervals, for the benefit of visually impaired audiences. For movies, this presents notable challenges – AD must occur only during existing pauses in dialogue, should refer to characters by name, and ought to aid understanding of the storyline as a whole.

To this end, we develop a new model for automatically generating movie AD, given CLIP visual features of the frames, the cast list, and the temporal locations of the speech; addressing all three of the 'who', 'when', and 'what' questions: (i) who – we introduce a character bank *consisting of the character's name, the actor that played the part, and a CLIP feature of their face, for the principal cast of each movie, and demonstrate how this can be used to improve naming in the generated AD; (ii) when – we investigate several models for determining whether an AD should be generated for a time interval or not, based on the visual content of the interval and its neighbours; and (iii) what – we implement a new vision-language model for this task, that can ingest the proposals from the character bank, whilst conditioning on the visual features using cross-attention, and demonstrate how this improves over previous architectures for AD text generation in an apples-to-apples comparison.*

1. Introduction

For in acts we must take note of ***who*** *did it, by what aids or instruments he did it (with),* ***what*** *he did, where he did it, why he did it, how and* ***when*** *he did it.* *Thomas Aquinas*

Audio Description (AD) is the descriptive narration of visual elements in a video, that are not represented in the original audio track. While there has been a proliferation of online content with *closed captioning*[1] due to advancements in ASR, a vast majority of video online does not have AD, mostly due to the prohibitive cost of generating it ($30 per minute[2]). Generating AD automatically at scale has multiple benefits; not only does it improve access for the visually impaired – it may also enhance the visual experience for sighted users (sight-free multitasking such as driving, enhanced memory for visual details, language learning, and also aiding those with other cognitive disabilities) [41]. Generating AD for movies is also an important research area in computer vision as it requires a system to perform multi-modal reasoning of long videos over time.

Figure 1. **AutoAD II:** We propose an automatic AD system that addresses key challenges - *when* to generate AD, *who* is in the scene and *what* is happening visually.

Despite these benefits, the progress in generating AD is still at a very nascent stage, due to the following challenges: (a) An ideal AD generation system should perform two tasks simultaneously – first, determine when to generate AD by proposing temporal segments; second, generate AD for the proposed segments. Previous works ignore the *when* completely, operating on already trimmed video segments [62]. (b) Secondly, given the strong relevance of characters to stories [27, 53], AD typically includes references to a character's name (*who* is in the scene), their emotion, and their actions. This is particularly challenging as characters change from movie to movie. Due to anonymised test sets (LSMDC [45]), the relevance of character names in AD is often ignored [62]. (c) Finally, AD also differs significantly from image or video captioning [33, 35, 46] in that it does not need to provide descriptions of events that can be understood from the sound track alone (such as dia-

†: also at Google Research
[1]Transcription of the speech

[2]https://www.3playmedia.com/blog/select-audio-description-vendor/

logue and ambient sounds) and should incorporate previous context to create a pleasurable listening experience without being repetitive or redundant. Such aspects require reasoning over multi-modal inputs (i.e., vision, text, and speech) over time while determining *what* to generate. In this work we propose an AD system that focuses on all these three W's – *when*, *who* and *what* (Fig 1).

To address *when*, we introduce a module to first propose temporal segments for AD. The time intervals for possible AD are constrained in that they do not overlap with the dialogue, but whether an AD is provided or not in the permissible time intervals depends on a number of factors including: the importance of the visual content to the story line, ambiguity in the audio soundtrack, and new information relative to previous AD.

For *who*, we introduce an AD model that can incorporate character information *on-the-fly* by referring to a text-visual *character bank* for that movie. One of the challenges of AD is that each movie has a different set of characters (and the actors that play them) that ought to be referenced in the AD captions. We address this by training a visual-language model to refer both to the external character bank and to the visual content of the scene when generating AD. The model can then be applied to any movie, given its cast list, without requiring retraining. This significantly improves references to characters, both in actual naming and in pronouns, in the generated AD compared to previous methods [18] that could only access names and pronouns present in the dialogue. Since character references appear in approximately 40% of AD, this is an important improvement.

The final challenge is *what* to generate, and involves reasoning over multimodal inputs – images, character bank and previous AD context. We do this via a novel multimodal cross-attention architecture, which ingests proposals from the character bank, and then conditions on visual features extracted from the movie frames.

Our contributions are the following. (1) We introduce a *Character Bank* to enable our AD generation model to label the characters appearing in the film. (2) We propose a Flamingo-style [1] architecture for the task, and compare this approach to the prompt style [36] architecture used previously for AD [18]. (3) We build a model for predicting *when* AD should be inserted, i.e. where on the timeline (using speech detection and visual cues). (4) Given the existing challenges with captioning based metrics [16], we employ a new evaluation metric for the AD content performance based on retrieval compared to other AD sentences in the movie. (5) We significantly outperform the previous state-of-the-art on the MAD dataset [18, 50].

2. Related Work

Dense Video Captioning. Dense video captioning is the task of temporally localising and captioning all events in an untrimmed video [26, 56, 66]. This differs from standard video captioning [33, 35, 46, 47], where the goal is to produce a single caption for a given trimmed video clip. While most methods for dense video captioning [23, 24, 26, 55, 57] consist of a 2-stage pipeline: a temporal localization stage followed by an event captioning stage; recent works [10, 11, 12, 31, 37, 44, 48, 49, 55, 56, 61, 66] jointly train the captioning and localization modules in order to improve inter-event relationships. The datasets for this task are largely obtained from web videos (*e.g.* YouCook2 [65], ViTT [19] and ActivityNet Captions [26]). Unlike these works, AD captions must be complementary to the audio information, tell a coherent story, and must not overlap with dialogue.

Movie Understanding. Early pioneering works exploit movies to learn actions [28]. The LSMDC [45] movie dataset sources its annotation from AD narrations and applies significant post-processing – character name anonymization and manual timestamp refinement – to ensure high correspondence between the short video clips and their captions. A series of short-form video tasks have since derived from LSMDC, including retrieval [4], person grounding [63], and sequential video captioning. TPAM [62] tackles the latter, prompting a frozen GPT-2 with local visual features. Later works propose tasks that require more long-form modelling, including aligning movies to books [52, 67] and synopses [60]; long video retrieval with the Condensed Movies Dataset (CMD) [3] and summarization [40].

Characters in Movies. A distinctive characteristic of movie understanding, setting it apart from other video domains, is its *character-centric* nature. Thus, character recognition is a prerequisite for the task, and many works have proposed automatic identification pipelines using face, voice, and body information [8, 15, 39, 51, 53]. Similar to our work, [7, 20, 39] initialize their character recognition pipeline with actor portraits, which can be further refined with noisy image captions [22]. Recently, CLIP has proved to be effective for zero-shot frame-level character labelling [25], alleviating the need for complex detection pipelines, which also inspires our character identification pipeline from CLIP features. Dense labelling of characters in movies and TV shows enables the modeling of interactions, relationships, and intentions – which can be formulated into classification [27], question answering [29], or captioning [30] tasks. Unlike these works, we use a character bank in a zero-shot practical setting for a real-world task: automatic AD generation.

Automated Audio Description. Visual captioning for assistive technologies is a growing area of computer vision research [5, 14, 17]. Yet, generating AD for video is still a relatively unexplored area of research. Initial work [58] applies heuristic cost-based filtering to video captioning on ActivityNet to generate diverse and relevant captions more akin to AD. In our earlier work [18], where we introduced the problem of AutoAD for movies, we provided a text-only

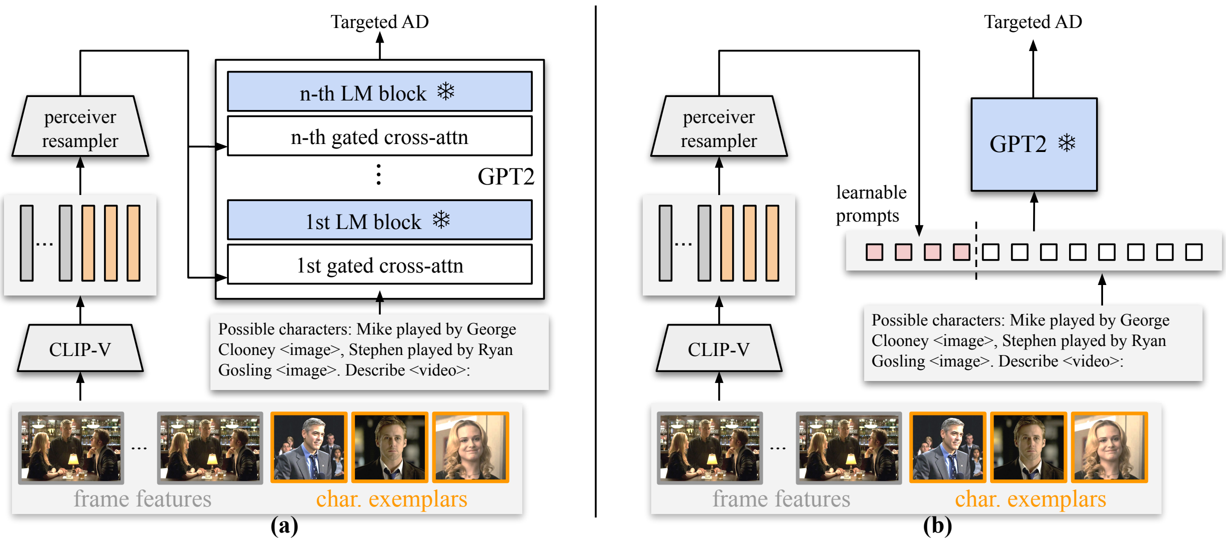


Figure 2. Architecture comparison: **(a) x-attn based method** vs. **(b) prompt learning based method**. We use architecture **(a)** in this paper. The character information is fed to the model in text form. Following Flamingo [1], we add text tags '⟨image⟩' to indicate the association between texts and character exemplar features. The model can also take context AD as additional text input, by simply appending more input text tokens.

AD corpus from over 7k movies available from the AudioVault website and used this for in-domain LLM pretraining. This resulted in substantial improvements to AD generation. We also use this dataset in this paper. However, our earlier work did not tackle the problem of *when* to generate AD, assuming these segments are given a-priori, nor did it deal with the problem of *who* – with the AD model failing to generate coherent character names, a critical component to story-coherent AD generation for long-form video content such as movies and TV shows. We address this failing in this paper.

3. New Models for Generating AD

As in [18], our method consists of adapting a large language model (LLM) for the task of generating AD. In the following sections, we describe three novel contributions: the first involves visual conditioning of multiple layers of the LLM (Section 3.1) in order to generate AD within a given time segment; the second describes a novel mechanism for incorporating character information *on-the-fly* that enables the model to infer a character's name in the scene (Section 3.2); and the third presents a simple approach for proposing temporal segments for where in time (when) the AD should be generated (Section 3.3).

3.1. A Visually Conditioned LM for Generating AD

Given a movie clip consisting of multiple frames $\mathbf{x}_i = \{\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_N\}$, our aim is to produce AD text $\mathcal{T}_i$ that describes the visual elements in a way that helps the visually impaired follow the story line. To achieve this we build on the capabilities of a pre-trained and frozen generative language model (LM). Broadly, two types of architecture are currently used to condition a LM on visual inputs: (a) by introducing additional layers into the LM that cross-attend to the visual input (examples include Flamingo [1]); or (b) by mapping the visual input to tokens that act as prompts for the LM (examples include ClipCap [36]). In both cases the LM is then able to generate descriptions of the visual inputs. In our case we have multiple video frames (represented by CLIP [42] vectors) and we use a Perceiver resampler to produce a fixed sized sequence of vectors for the visual input. The two types of architecture are illustrated in Figure 2.

In this paper, we develop a model based on type (a), with additional cross-attention layers in the LM. We describe this in more detail below, and demonstrate in the results that it has superior performance over type (b) in our case. We also briefly discuss the advantages and disadvantages of the two types of architecture below.

Architecture description. In detail, the architecture has three components: (i) a CLIP encoder that generates visual features from the input movie frames as $\mathbf{z} = f_{\text{CLIP}}(\mathcal{I}_1, \mathcal{I}_2, ...)$; (ii) a Perceiver resampler that models the contextual information amongst these visual features and summarizes them into a sequence of fixed-length vectors: $\hat{\mathbf{x}} = \mathcal{P}([\mathbf{z}; \mathbf{x}])$, where $\mathbf{x}$ are learnable latent states of the Perceiver module $\mathcal{P}$; and (iii) trainable cross-attention blocks that are inserted into the frozen language model. Each cross-attention block is controlled by a `tanh` gating mechanism, which is initialized with zero values such that the language model maintains its original activation at the beginning of the training as $\mathbf{h}_{j+1} = \mathbf{h}_j + \texttt{tanh}\big(\texttt{XAttn}(\mathbf{h}_j, \hat{\mathbf{x}}, \hat{\mathbf{x}})\big)$, where $\mathbf{h}_j$ is the hidden vector of the j-th block of the language model and $\texttt{XAttn}(q, k, v)$ denotes the cross-attention module with its query, key and value inputs in order.

Flexibility for multimodal context. For our purposes the Flamingo-like architecture offers flexibility: the input can simply be the video frames (via the Perceiver resampler) and a text prompt to the LM, such as 'Describe ⟨video⟩:' to start the AD generation. However, in the case that additional image and text context is available (as in the additional character naming and image examples from the character bank, described below), then this can simply be prepended to the prompt, and the trainable cross-attention layers learn how to correctly attend to both the video frames and the image examples

In contrast, for the second type of architecture where the visual input acts as a prompt to the LM, it is necessary to train new tokens, such as BOS [18], in order to separate visual prompts from text prompts and start the AD generation.

In summary, both architectures build on frozen LMs (previous works [1] show that finetuning an LLM on the task-of-interest can harm their generalization) and have trainable parameters to allow the LM to condition on the visual input and adapt to the AD task. However, the cross-attention type of architecture offers greater flexibility and, as will be seen, superior performance.

3.2. Incorporating a Character Bank

Our goal is to recognize *active* characters – defined as those appearing on-screen – in a given movie clip by leveraging the movie cast list from an external movie database $\mathcal{M}$, and thereby provide the information about active characters to the AD generation. To this end, we (i) build visual character exemplar features by exploiting actor portrait images from $\mathcal{M}$, further calibrated by comparing against the movie frames, and (ii) train a character recognition module that predicts the active characters given their exemplars and the movie clip.

Given a long-form movie $\mathcal{V}$, the corresponding cast list can be queried from the database $\mathcal{M}$. The character bank for this movie $\mathcal{V}$ can be written as $\mathcal{B}_{\mathcal{V}} = \{[\texttt{char}_j, \texttt{act}_j, \mathcal{A}_j]\}_{j=1}^{C}$, where C denotes the number of characters, $\texttt{char}_j$ is the character name in the movie, $\texttt{act}_j$ is the actor name, and $\mathcal{A}_j$ is the actor's portrait image from the movie database. Below are two example items in a character bank:

$$\{[\text{Jack Dawson, Leonardo DiCaprio}, \mathcal{A}_{\text{LD}}],$$
$$[\text{Rose DeWitt-Bukater, Kate Winslet}, \mathcal{A}_{\text{KW}}], ...\}$$

Calibrating the actor portrait feature. An actor's portrait image can differ considerably in appearance from the character in the movie due to various factors, such as hairstyle, makeup, dress, ageing, or camera viewpoint [39]. In particular, for older movies with different dressing styles and fewer close-up shots, actor portraits might lie very far from the movie's frame in the feature space. To overcome this issue, we propose a calibration step. Instead of using the image features from the actor's portrait, we retrieve the top-k nearest frames within the same movie, and average

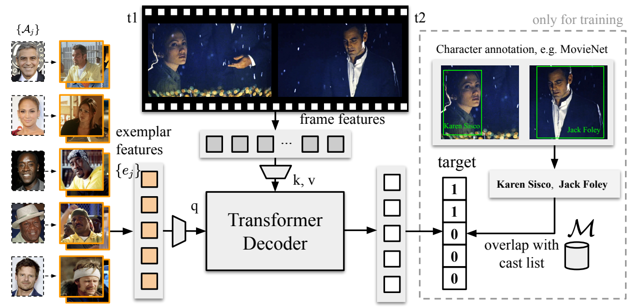


Figure 3. **Character recognition module**: Given character exemplar features for C characters $\{e_j\}_{j=1}^{C}$, and movie frame features for a given clip, we formulate a binary classification problem to determine whether each character is active in the scene or not. From left to right: portrait images determine a within-movie exemplar for each character, and each of these exemplars provides a query to the transformer. The frame features provide the keys and values for the transformer. The output of each query is used to determine the binary question of whether that character is in the clip or not. The module is trained using MovieNet data where characters are annotated for the frames of a clip. The binary labels are formed by checking the MovieNet character annotations against the cast list in our movie database.

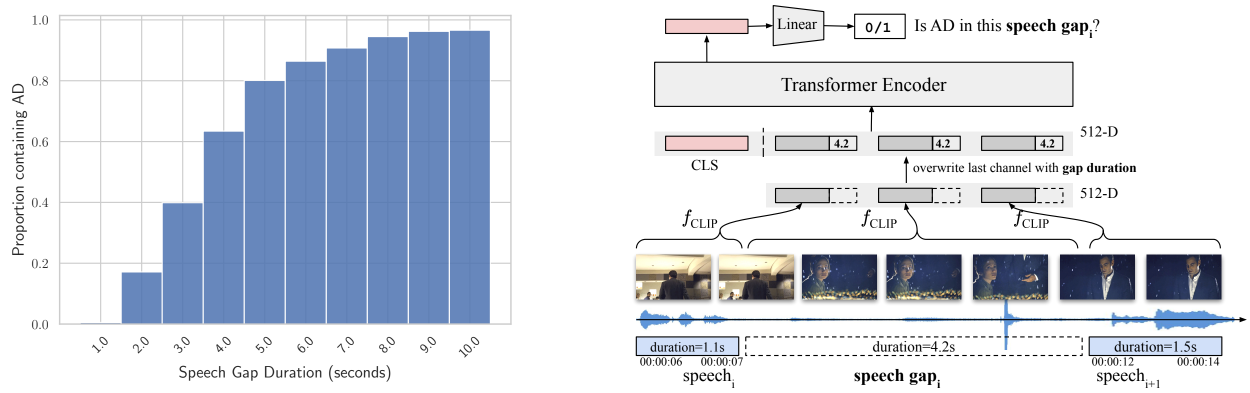
the frame features to create an exemplar for that character. Specifically, let $\mathbf{z}_{\mathcal{V}} = f_{\text{CLIP}}(\mathcal{V})$ denote the sequence of visual features of the movie $\mathcal{V}$, and given a portrait image of actor j as $\mathcal{A}_j$, we first compute its visual feature $z_j = f_{\text{CLIP}}(\mathcal{A}_j)$, and compare it against $\mathbf{z}_{\mathcal{V}}$ via cosine similarity. The character exemplar feature of actor j in the movie $\mathcal{V}$ can then be computed by:

$$e_j = \frac{1}{k} \sum \mathbf{z}_{\mathcal{V}} \left[\text{top-}k \left(\frac{z_j^{\top} \mathbf{z}_{\mathcal{V}}}{|z_j| \cdot |\mathbf{z}_{\mathcal{V}}|} \right) \right],$$

where top-k finds the indices of the k most similar frames and $[.]$ symbol means the indexing operation. In the Appendix, we show this calibration procedure (i.e., replacing z_j with e_j) is essential for constructing reliable character banks.

Recognizing characters in the movie clip. Not all characters appear on-screen at the same time. With a character bank $\mathcal{B}_{\mathcal{V}}$ for the movie $\mathcal{V}$, our goal is to recognize the *active* characters that appear between times t_1 and t_2 to enable naming them in the AD generation. This character recognition task may be achieved by face detectors [13], or even speaker recognition from voice [59]. However, for the movie datasets used in this work, the absence of raw frames prohibits the use of face detection, and the characters mentioned in AD may not necessarily be speaking. Instead, we propose to use a character recognition module based purely on frame-level visual features and the character bank information $\mathcal{B}_{\mathcal{V}}$.

As shown in Fig. 3, both the exemplar features for each character $\{e_j\}_{j=1}^{C}$ and the movie frame features $\mathbf{z}_{\mathcal{V}[t_1,t_2]}$ are



(a) (b)

Figure 4. **(a) Proportion of speech gaps containing AD relative to their duration** – very short speech gaps rarely contain AD, and large speech gaps nearly always contain AD. The statistics are from the MAD training set. **(b) Architecture for AD temporal proposal classification.** Given a speech gap, the model classifies whether or not AD should be inserted in the gap, taking visual and duration cues as input.

first fed to a linear projection layer, which aims to project general visual features onto a face feature space [25]. Then a relatively shallow (2-block) transformer decoder takes both projected features and outputs a probability for each character on whether they appear between times $[t_1, t_2]$. This module is trained with a binary classification loss. The labels for training are obtained from face annotations available in the MovieNet [21] dataset. In Sec. 5.2.2, we compare our model with a baseline of simply thresholding the similarity between the character's exemplar and movie frame features with a scalar α. The experiment shows that the transformer decoder module outperforms this baseline. In the Appendix, we also experiment with training from labels obtained by running named entity recognition (NER) [38] on the AD annotation, and show that this performs worse than using labels obtained from MovieNet.

Other approaches. In our earlier work [18], we mined character names from subtitles using NER and provided these as prompts to the AD generation model – but this failed to reference character names effectively. The failure may be because character names occur sparsely in subtitles (Based on the MAD-train movies, approximately 13% of subtitle sentences contain character names, compared to 41% of AD, see Appendix Table A.1 and Table A.2) or because the names found may refer to off-screen characters. In Figure 5, we show that the movie cast list (when narrowed down to those that appear in the scene with our character recognition module) provides high precision and recall of active on-screen characters.

Using the character bank for AD generation. A trained character recognition module can recognize the *active* characters in any video clip $\mathcal{V}_{[t_1,t_2]}$. Next, we feed this character information into our AD generation pipeline.

In Sec. 3.1, we introduce a versatile cross-attention-based architecture which supports textual and other multi-modal inputs. We feed in character information to the model mainly by *text prompting*. In more detail, given a character list for the movie clip $\mathcal{V}_{[t_1,t_2]}$, we explore three different ways of supplying the active characters in the scene. Let's assume [$\texttt{char}_1$, $\texttt{char}_2$] are recognized as active. The prompting templates are then:

1. "possible characters: $\texttt{char}_1$, $\texttt{char}_2$."
2. "possible characters: $\texttt{char}_1$ played by $\texttt{act}_1$; $\texttt{char}_2$ played by $\texttt{act}_2$."
3. "possible characters: $\texttt{char}_1$ played by $\texttt{act}_1$ $\langle$image$\rangle$; $\texttt{char}_2$ played by $\texttt{act}_2$ $\langle$image$\rangle$."

Note that in method (3), the $\langle$image$\rangle$ tag is purely in the text form; therefore, in this setting, we feed in the character exemplar features $[e_1, e_2]$ in the corresponding order to the perceiver resampler, such that it can learn the association between the character's identity and the movie clip.

3.3. Proposing AD Temporal Segments

An ideal AD system must not only generate high quality AD narrations (*what*), but must also decide *when* to generate AD. The Web Content Accessibility Guidelines 2.0 [9] outlines specific criteria for successful AD: (i) it must only be added during existing pauses in dialogue; and (ii) it need not be added when all of the video information is already provided in existing audio.

In practice, long pauses in dialogue and the subjectivity of the second guideline mean these provide rather weak constraints on the timing of AD, resulting in large variations between human-generated AD timestamps for the same movie[3]. Such variation makes it difficult to learn and evaluate fine-grained model predictions of proposed AD temporal segments. Therefore, we formulate the temporal proposal task into one of binary classification: *given*

[3]An analysis on Audio Descriptions and inter-annotator agreement is provided in the Appendix.

an existing pause in dialogue, should AD be inserted in the pause? This coarse-grained formulation has much higher inter-annotator agreement and inherently satisfies the first guideline for generating AD[3].

Given a long-form movie $\mathcal{V}$, our goal is to identify inactive speech regions and classify whether or not they should contain audio description (AD). Thirty second intervals are extracted from the movie. Automatic speech recognition (ASR) is applied to the audio stream A of each interval to obtain a set of speech segments $S = s_1, ..., s_N$, where each $s_i = (t_i^0, t_i^1)$ indicates the start and end times of a speech utterance. The text from the ASR segments s_i are given as input to a BERT encoder, prefixed and suffixed with discrete timestamp tokens $\tau \in$ <|t00|>, ..., <|t60|> denoting their start and end times rounded to 0.5 seconds. The gaps between utterances are represented by inserting <|mask|> tokens between timestamped segments. In addition to speech, we also provide visual features from CLIP [43], sampled every second from the context window, and append these to the input text sequence.

A binary classification head is then applied to each <|mask|> token to predict whether the gap should contain AD. The model is trained end-to-end using binary cross-entropy loss. At inference time, this model is applied in a sliding window fashion to the full movie $\mathcal{V}$. An overview of the method is provided in Figure 4(a). Further details are provided in Appendix C.4.

By analysing the distribution of AD data (Figure 4(a)), we find that whether or not AD is contained within a given speech gap is highly correlated with the duration of said gap. In fact, gaps of two seconds or less contain AD only 17% of the time. At the other extreme, gaps of 6 seconds or more contain AD 85% of the time. Due to such strong duration correlations, we restrict the prediction task to speech gaps between two and five seconds. The classification of whether to insert AD within shorter or longer speech gaps can be obtained via a hard-coded rule. The effect of timestamp tokens and visual features are given in Section 5.3.

4. Implementation Details

4.1. Training Data

MAD [50] is a movie audio description dataset consisting of movie frame features and timed AD in the text form. We follow [18] and use 488 movies as the training set. Specifically for AD, we use the same preprocessing pipeline proposed in [18] to obtain high-quality ASR outputs. We use the 'named' version of MAD dataset. **AudioVault-AD [18]** is a text-only corpus of AD for 7057 movies downloaded from the AudioVault website. The movies are not included in MAD dataset. We use the AudioVault-AD for text-only pretraining. **WebVid [4]** is a dataset of 2.5M captioned short videos for visual-only pretraining. We find the NER from both LSMDC-train and MAD-train contain non-trivial noise, despite the one for LSMDC-train having been manually verified. **MovieNet** [21] is a movie dataset providing movie keyframes and various annotations including character names for each keyframe. We choose an overlap of MovieNet movies with MAD training movies to train the character recognition module.

4.2. Testing Data

MAD-eval [18] consists of 10 movies for evaluating AD captioning from the LSMDC validation and testing set. The timestamps from LSMDC are manually edited to ensure high visual correspondence with the caption. We treat this as our standard evaluation for measuring AD caption quality.

MAD-t-eval is our proposed benchmark for evaluating AD time point prediction. The edited timestamps in *MAD-eval* are not appropriate for measuring temporal proposals because they are expanded and often overlap with speech segments. Therefore we evaluate time prediction models on *MAD-t-eval*, consisting of three movies (from MAD-eval) where the AD and their original timestamps are sourced from Audiovault and manually verified. We restrict the evaluation to speech gaps with a duration between two and five seconds, resulting in 530 gaps across the three movies.

4.3. Collecting Character Banks

The character information for movies can be collected from online databases or review websites like IMDb[4]. In detail, for each movie in Audiovault, MAD-train and the MAD-eval datasets, we download the top 10 cast information from IMDb including the actor names, their character role name, and the actor portrait image. Full details are provided in Appendix.

4.4. Training & Inference Recipe

In this section, we first outline the architectures used for each module in the AD captioning system; we then describe how each module individually is pretrained; and finally we describe the finetuning and inference details for the full AD captioning system.

4.4.1 Architectural components

We give a summary here with fuller details in the Appendix.

AD generation model (Section 3.1) is built on top of *GPT2-small*, specifically the open-source version from HuggingFace. We insert an X-attn block after *each* of the transformer block of GPT-2. The perceiver resampler has two transformer decoder blocks with 10 latent vectors. For the visual encoder, we use CLIP [42] ViT-B/32 model which extracts 512-d features for each movie frame. These features are provided by the MAD dataset [50].

Character recognition module (Section 3.2) consists of a linear layer and a 2-block transformer decoder. It takes the

[4]https://www.imdb.com/

movie character exemplar features $\{e_j\}$ and movie clip features as input, and outputs a probability for each exemplar feature.

AD temporal proposals (Section 3.3). For VAD, we use the *pyannote* model [6]. For the temporal proposal classification model, we use a 3-layer transformer encoder with sin-cos positional embeddings. For the visual features we use CLIP ViT-B/32 [42].

4.4.2 Pretraining recipe

To overcome the limited amount of paired AD training data, we follow [18] and perform partial data pretraining for each component in our modular architecture.

GPT-2 (Section 3.1). We follow [18] and perform secondary in-domain pretraining of GPT-2 on the Audiovault text-only corpus to match the text distribution for AD generation.

Video captioning (Section 3.1). We pretrain the cross-attention visual captioning blocks on 2.5M video-text pairs from WebVid [4], while keeping the GPT-2 LM block weights frozen.

Character recognition module (Section 3.2). The module is trained on character name labels from MovieNet [21].

4.4.3 Finetuning & Inference

AD captioning (Section 3.1). With the recognized active character list as an additional input and model parameters partially pretrained, the AD generation model is finetuned on MAD-train with an AdamW [34] optimizer and 10^{-4} learning rate. For output text sampling, we use beam search with the beam size of 5 and report results by the top-1 beam-searched outputs, since it performs slightly better than greedy search on multiple scenarios. The full training details are in Appendix.

AD temporal proposals (Section 3.3). The pretrained BERT (base-uncased) is finetuned on the MAD dataset for three epochs, with a BCE loss and an AdamW [34] optimizer of learning rate 10^{-4}. The classification task at training and inference is restricted to speech gaps with durations between 2-6 seconds.

5. Experiments

The experimental section is organised as follows: we start by describing the evaluation metrics in Section 5.1; then in Section 5.2, we demonstrate the effectiveness of our proposed architecture and training strategy, based on the groundtruth AD time segments, for example, visually conditioned LM, effect of character bank, and partial-data pretraining; in Section 5.3, we evaluate on the temporal proposal, and present qualitative results in Section 5.4.

5.1. Evaluation Metrics

Classic metrics for text generation. We adopt classic captioning metrics to compare the generated AD to the ground-truth AD, namely, ROUGE-L [32] (**R-L**) and CIDEr [54] (**C**).

Retrieval-based metric for text sequence generation. We propose a new recall-based metric: 'Recall@k within N neighbours' (**R@k/N**). In detail, given two sequences of generated texts and ground-truth (GT) texts in their temporal order, for each generated text at time point $[t_1, t_2]$, we compute the Recall@k with N adjacent GT texts, then average the score. To compute recall, we use the BertScore [64] as the text similarity measure. There are two benefits of this metric: (i) Classic captioning metrics like CIDEr or ROUGE-L are mainly based on n-gram accuracy, which tends to over-penalise the system on linguistic text variations, *i.e.* because there are multiple ways to express the same meaning. The retrieval-based **R@k/N** metric is less affected by these low-level variations in the text. (ii) the metric is operated within a window of N neighbouring texts along the time axis, which considers the arbitrary text positioning of long sequence captioning.

Metrics for character recognition and time proposal. The character recognition (described in Section 3.2) and the time segment proposal tasks (described in Section 3.3) are formulated as multi-label and binary classification problems respectively We report ROC-AUC and Average Precision for the classifiers, with class macro-averaging for the multi-label case.

5.2. Audio Description on GT segments

This section focuses on the effectiveness of each proposed component in the AD generation pipeline, based on the *ground-truth* AD time segments, as shown in Table 1.

5.2.1 Architecture comparison

We investigate two ways for conditioning a pre-trained and frozen generative language model (LM) with visual inputs, that is, (a) by introducing additional layers into the LM that cross-attend to the visual input, or (b) by mapping the visual input to tokens that act as prompts for the LM. Comparing rows 'B1 vs A1' and 'B4 vs A2' in Table 1, the architecture with newly introduced cross-attention outperforms the prompting-based architecture both with or without the character bank inputs. The performance gain comes from greater interaction between visual and textual features by its interleaved design.

5.2.2 Effect of character bank

Here, we start by investigating the effect of incorporating the character bank in three different ways as discussed in Section 3.2, followed by comparing our proposed character recognition module with a naïve baseline.

Exp.	AD Context	PT	Arch	CharBank Settings Source	Char.	Act.	Exem.	R-L	C
A1	✗	✗	Prompt	-	✗	✗	✗	9.3	6.7
A2	✗	✗	Prompt	recog.	✓	✓	✓	10.4	11.0
B1	✗	✗	X-Attn	-	✗	✗	✗	9.7	10.0
B2	✗	✗	X-Attn	recog.	✓	✗	✗	10.8	14.2
B3	✗	✗	X-Attn	recog.	✓	✓	✗	11.1	15.0
B4	✗	✗	X-Attn	recog.	✓	✓	✓	12.7	18.3
B5	✗	✗	X-Attn	full-cast	✓	✓	✓	10.9	14.9
C1	✗	AV&WV	X-Attn	recog.	✓	✓	✓	13.1	19.2
C2	✓(recurrent)	AV&WV	X-Attn	recog.	✓	✓	✓	13.4	19.5

Table 1. **Ablations for AD generation.** We ablate the effect of the cross-attention module and character bank, and show the effect of partial-data pretraining. All models are trained on MAD-train-`named` and evaluated on MAD-eval-`named`. Performance is reported in terms of ROUGE-L (R-L) and CIDEr (C).

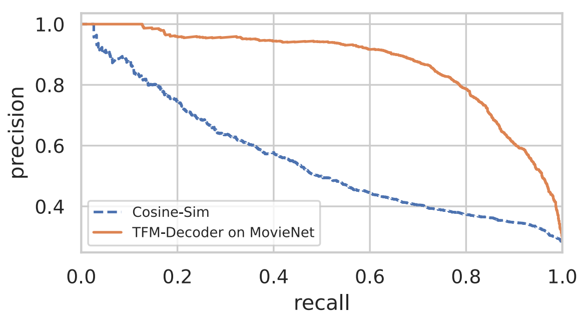


Figure 5. The Precision-Recall curve for the character recognition methods, computed on 4 MAD-eval movies that have character annotations from MovieNet. We compare two methods: thresholding actor-movie cosine similarity, and learned transformer decoder on MovieNet. The precision/recall is calculated on a per-character basis, *i.e.* the precision/recall of the cosine thresholds to correctly find a character name mentioned in the AD. More baselines are described in the Appendix.

Methods	ROC AUC	Average Precision
Cosine-Sim	0.72	0.55
TFM Decoder	**0.93**	**0.87**

Table 2. We compare different methods for recognising characters in a clip, reported on 4 MAD-eval movies that have character annotations from MovieNet.

Choices for exploiting character bank. By default, the model takes the predicted 'active' characters in the scene from the character recognition module. From the comparison rows 'B1-4', we can draw the following observations: (i) injecting character names gives a clear performance gain ('B2 vs B1'), highlighting the dependency of the AD task on character names; (ii) inputting additional actor names only brings marginal improvements ('B3 vs B2'), we conjecture this is because CLIP have seen large number of celebrities' picture-names pairs at pre-training stage, the visual features have thus already encoded such information; (iii) feeding in characters' exemplar features improves the performance ('B4 vs B3'), showing the complementary nature of visual-textual features; (iv) presenting the full cast list (*i.e.* all 10 characters downloaded from IMDb) as character bank leads to inferior performance, as shown by comparison between B4 and B5. This is because the full cast list introduces irrelevant characters to the AD generation pipeline and harms the training, especially with a limited size of training data. This comparison also shows the necessity and effectiveness of our character recognition module, which provides a higher-quality character list to aid AD generation.

Methods	ROC AUC	Average Precision
Baseline (Duration)	0.70	0.53
TFM (Visual)	0.71	0.52
TFM (Visual+Duration)	**0.78**	**0.61**

Table 3. Results on the binary AD temporal proposal task on the `MAD-t-eval` benchmark where TFM refers to the transformer encoder architecture.

Character recognition module. We compare the proposed character recognition module (described in Fig. 3) with the baseline method that simply thresholds CLIP similarities between exemplar features and frame features. Specifically, we train the character recognition module on 550 MovieNet movies with character annotations, and report results on 4 MAD-eval movies that have character annotations from MovieNet. Since the task of recognizing active characters is a binary classification, we report ROC-AUC and Average Precision, as shown in Table 2, our proposed character recognition module clearly outperforms the baseline by a large margin. More details and discussion on a PR curve are provided in the Appendix.

5.2.3 Partial-data pretraining and context

Following [18], we also pre-train our model with partial-data, for example, AudioVault and WebVid, as well as incorporate previous AD as context for the model. The results show that AD generation can be further improved by combining these methods, showing that our newly introduced cross-attention module and character bank are orthogonal to the contributions in [18].

5.3. Temporal proposal results

In Table 3 we compare the different time proposal classification models with the baseline threshold method. We see that training a transformer encoder architecture with both visual and speech gap duration information as inputs brings substantial improvements (0.53 AP to 0.61 AP)

5.4. Qualitative Results

Fig. 6 shows four qualitative examples. It shows that the character recognition module is able to recognize active

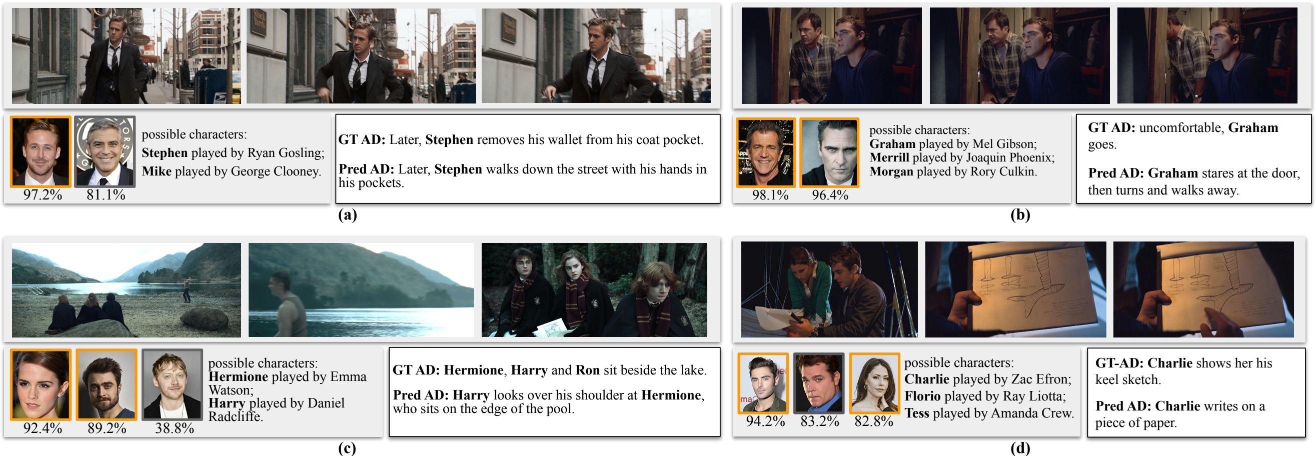


Figure 6. Qualitative results of our method. For a given movie clip, the character recognition module can recognize active characters on-the-fly and its results are fed into the AD generation pipeline. Note that for character recognition, we simply threshold the 'active' characters in the scene with a probability of 50%. The probability shown below characters' portraits is the output of our recognition module, with correctly recognized characters marked using an orange border. For visualization purpose, we show the character's IMDb portrait image but the model actually takes in exemplar features as input. Three frames are shown for each movie clip. To illustrate the effect of character bank, this model is trained *without* context AD – that is, naming information is only from the character bank. The movies are from: **(a)**: Ides of March (2011), **(b)**: Signs (2002), **(c)**: Harry Potter and the Goblet of Fire (2005), **(d)**: Charlie St. Cloud (2010).

characters reasonably well, and the AD generation module can associate the active characters with the descriptions. Note that even if the character recognition module proposes incorrect characters, the AD generation pipeline has learned to *ignore* such irrelevant characters for AD generation, such as 'Mike' in sample **(a)** and Morgan in sample **(b)**. Given that a large portion of AD sentences (41%) contains human identity like those samples, recognizing characters is an essential capability for high-quality AD generation. More examples in the Appendix.

5.5. Comparison with state-of-the-art

In Table 4, we report AD captioning results on the MAD-eval benchmark and achieve state-of-the-art performance by considerable margins across both the local and recurrent settings. Note that our method *without* context AD or AV/WebVid pretraining already surpasses AutoAD-I (CIDEr 18.3 vs 14.3). Adding partial data pretraining on AV/WebVid and context AD further increases the performance (CIDEr 19.5 vs 14.3). Note, one issue that affects the evaluation is that although we might correctly identify a character, and they are referred to in the AD, the actual name may differ since a character may be named in a multitude of ways, *e.g.* first name only – 'Albus', or last name with a prefix – 'Mr. Dumbledore', their professions, titles or pronouns – 'Professor', 'Prof. Dumbledore' or 'He', their relationships to other characters – 'Aberforth's brother', or other nicknames etc. We leave the resolution of this issue to future work.

Methods	Time window	Pretrain Data	R-L	C	R@5/16
ClipCap [36]	local	CC3M	8.5	4.4	36.5*
AutoAD-I [18]	local	WebVid	9.9	10.0	38.2*
AutoAD-I [18]	local	AV & WebVid	10.3	12.1	39.8*
Ours	local	None	12.7	18.3	45.6
Ours	local	AV & WebVid	**13.1**	**19.2**	**51.3**
AutoAD-I [18]	recurrent	AV & WebVid	11.9	14.3	42.1*
Ours	recurrent	AV & WebVid	**13.4**	**19.5**	**50.8**

Table 4. Comparison with other methods on `MAD-eval` benchmark under both the local (without AD context) and recurrent (with previously predicted AD as context) settings. *Denotes results re-implemented by us using the same evaluation setting.

6. Discussion and Future Work

Taken together this paper has proposed all the elements needed for a fully automated AD system: when to produce AD, what it should contain, and who it should describe (naming). Note these sub-tasks can probably be done jointly by using a transformer decoder with special time tokens, such as Whisper [43] or Vid2Seq [61]. Predicting accurate timestamps for such architectures [2], modelling long-term dependency and leveraging multi-modal information are exciting challenges towards human-level movie understanding.

Acknowledgements. We thank the AudioVault team for their priceless contribution of Audio Description archives. This research is funded by EPSRC PG VisualAI EP/T028572/1, and ANR-21-CE23-0003-01 CorVis.

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022. 2, 3, 4, 14, 15

[2] Max Bain, Jaesung Huh, Tengda Han, and Andrew Zisserman. WhisperX: Time-accurate speech transcription of long-form audio. In *Interspeech*, 2023. 9

[3] Max Bain, Arsha Nagrani, Andrew Brown, and Andrew Zisserman. Condensed movies: Story based retrieval with contextual embeddings. In *Proc. ACCV*, 2020. 2

[4] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proc. ICCV*, 2021. 2, 6, 7

[5] Jeffrey P Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, et al. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the 23rd annual ACM symposium on User interface software and technology*, pages 333–342, 2010. 2

[6] Hervé Bredin, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill. Pyannote. audio: neural building blocks for speaker diarization. In *Proc. ICASSP*, pages 7124–7128. IEEE, 2020. 7

[7] Andrew Brown, Ernesto Coto, and Andrew Zisserman. Automated video labelling: Identifying faces by corroborative evidence. In *2021 IEEE 4th International Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 77–83. IEEE, 2021. 2

[8] Andrew Brown, Vicky Kalogeiton, and Andrew Zisserman. Face, body, voice: Video person-clustering with multiple modalities. In *ICCV 2021 Workshop on AI for Creative Video Editing and Understanding*, 2021. 2

[9] Ben Caldwell, Michael Cooper, Loretta Guarino Reid, Gregg Vanderheiden, Wendy Chisholm, John Slatin, and Jason White. Web content accessibility guidelines (wcag) 2.0. *WWW Consortium (W3C)*, 290:1–34, 2008. 5, 12

[10] Aman Chadha, Gurneet Arora, and Navpreet Kaloty. iPerceive: Applying common-sense reasoning to multi-modal dense video captioning and video question answering. In *Proc. WACV*, 2021. 2

[11] Shaoxiang Chen and Yu-Gang Jiang. Towards bridging event captioner and sentence localizer for weakly supervised dense event captioning. In *Proc. CVPR*, 2021. 2

[12] Chaorui Deng, Shizhe Chen, Da Chen, Yuan He, and Qi Wu. Sketch, ground, and refine: Top-down dense video captioning. In *CVPR*, 2021. 2

[13] Jiankang Deng, Jia Guo, Evangelos Ververas, Irene Kotsia, and Stefanos Zafeiriou. RetinaFace: Single-shot multi-level face localisation in the wild. In *Proc. CVPR*, 2020. 4

[14] Pierre Dognin, Igor Melnyk, Youssef Mroueh, Inkit Padhi, Mattia Rigotti, Jarret Ross, Yair Schiff, Richard A Young, and Brian Belgodere. Image captioning as an assistive technology: Lessons learned from VizWiz 2020 challenge. *arXiv preprint arXiv:2012.11696*, 2020. 2

[15] Mark Everingham, Josef Sivic, and Andrew Zisserman. "Hello! My name is... Buffy" – automatic naming of characters in TV video. In *Proc. BMVC*, 2006. 2

[16] Soichiro Fujita, Tsutomu Hirao, Hidetaka Kamigaito, Manabu Okumura, and Masaaki Nagata. Soda: Story oriented dense video captioning evaluation framework. In *Proc. ECCV*, pages 517–531. Springer, 2020. 2

[17] Danna Gurari, Yinan Zhao, Meng Zhang, and Nilavra Bhattacharya. Captioning images taken by people who are blind. In *Proc. ECCV*, pages 417–434. Springer, 2020. 2

[18] Tengda Han, Max Bain, Arsha Nagrani, Gül Varol, Weidi Xie, and Andrew Zisserman. AutoAD: Movie Description in Context. In *Proc. CVPR*, 2023. 2, 3, 4, 5, 6, 7, 8, 9, 12, 14

[19] Gabriel Huang, Bo Pang, Zhenhai Zhu, Clara Rivera, and Radu Soricut. Multimodal pretraining for dense video captioning. *arXiv preprint arXiv:2011.11760*, 2020. 2

[20] Qingqiu Huang, Wentao Liu, and Dahua Lin. Person search in videos with one portrait through visual and temporal links. In *Proc. ECCV*, pages 425–441, 2018. 2

[21] Qingqiu Huang, Yu Xiong, Anyi Rao, Jiaze Wang, and Dahua Lin. MovieNet: A holistic dataset for movie understanding. In *ECCV*, 2020. 5, 6, 7

[22] Qingqiu Huang, Lei Yang, Huaiyi Huang, Tong Wu, and Dahua Lin. Caption-supervised face recognition: Training a state-of-the-art face model without manual annotation. In *Proc. ECCV*, 2020. 2

[23] Vladimir Iashin and Esa Rahtu. A better use of audio-visual cues: Dense video captioning with bi-modal transformer. In *BMVC*, 2020. 2

[24] Vladimir Iashin and Esa Rahtu. Multi-modal dense video captioning. In *CVPR Workshops on Multimodal Learning*, 2020. 2

[25] Bruno Korbar and Andrew Zisserman. Personalised clip or: how to find your vacation videos. In *British Machine Vision Conference*, 2022. 2, 5

[26] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proc. ICCV*, pages 706–715, 2017. 2

[27] Anna Kukleva, Makarand Tapaswi, and Ivan Laptev. Learning interactions and relationships between movie characters. In *Proc. CVPR*, pages 9849–9858, 2020. 1, 2

[28] Ivan Laptev, Marcin Marszałek, Cordelia Schmid, and Benjamin Rozenfeld. Learning realistic human actions from movies. In *Proc. CVPR*, 2008. 2

[29] Jie Lei, Licheng Yu, Mohit Bansal, and Tamara L Berg. Tvqa: Localized, compositional video question answering. In *EMNLP*, 2018. 2

[30] Jie Lei, Licheng Yu, Tamara L Berg, and Mohit Bansal. TVR: A large-scale dataset for video-subtitle moment retrieval. In *Proc. ECCV*, pages 447–463. Springer, 2020. 2, 17

[31] Yehao Li, Ting Yao, Yingwei Pan, Hongyang Chao, and Tao Mei. Jointly localizing and describing events for dense video captioning. In *Proc. CVPR*, 2018. 2

[32] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004. 7

[33] Kevin Lin, Linjie Li, Chung-Ching Lin, Faisal Ahmed, Zhe Gan, Zicheng Liu, Yumao Lu, and Lijuan Wang. SwinBERT: End-to-end transformers with sparse attention for video captioning. In *Proc. CVPR*, 2022. 1, 2

[34] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 7

[35] Huaishao Luo, Lei Ji, Botian Shi, Haoyang Huang, Nan Duan, Tianrui Li, Xilin Chen, and Ming Zhou. Uni-ViLM: A unified video and language pre-training model for multimodal understanding and generation. *arXiv preprint arXiv:2002.06353*, 2020. 1, 2

[36] Ron Mokady, Amir Hertz, and Amit H Bermano. Clip-Cap: CLIP prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021. 2, 3, 9

[37] Jonghwan Mun, Linjie Yang, Zhou Ren, Ning Xu, and Bo-hyung Han. Streamlined dense video captioning. In *Proc. CVPR*, 2019. 2

[38] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007. 5, 16

[39] Arsha Nagrani and Andrew Zisserman. From benedict cumberbatch to sherlock holmes: Character identification in tv series without a script. In *Proc. BMVC*, 2017. 2, 4

[40] Pinelopi Papalampidi, Frank Keller, and Mirella Lapata. Movie summarization via sparse graph construction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13631–13639, 2021. 2

[41] Elisa Perego. Gains and losses of watching audio described films for sighted viewers. *Target*, 28(3):424–444, 2016. 1

[42] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proc. ICML*, 2021. 3, 6, 7

[43] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *OpenAI blog*, 2022. 6, 9

[44] Tanzila Rahman, Bicheng Xu, and Leonid Sigal. Watch, listen and tell: Multi-modal weakly supervised dense event captioning. In *Proc. ICCV*, 2019. 2

[45] Anna Rohrbach, Marcus Rohrbach, Niket Tandon, and Bernt Schiele. A dataset for movie description. In *Proc. CVPR*, 2015. 1, 2

[46] Paul Hongsuck Seo, Arsha Nagrani, Anurag Arnab, and Cordelia Schmid. End-to-end generative pretraining for multimodal video captioning. In *Proc. CVPR*, pages 17959–17968, 2022. 1, 2

[47] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Association for Computational Linguistics*, 2018. 2

[48] Zhiqiang Shen, Jianguo Li, Zhou Su, Minjun Li, Yurong Chen, Yu-Gang Jiang, and Xiangyang Xue. Weakly supervised dense video captioning. In *Proc. CVPR*, 2017. 2

[49] Botian Shi, Lei Ji, Yaobo Liang, Nan Duan, Peng Chen, Zhendong Niu, and Ming Zhou. Dense procedure captioning in narrated instructional videos. In *Association for Computational Linguistics*, 2019. 2

[50] 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 *Proc. CVPR*, 2022. 2, 6, 12

[51] Makarand Tapaswi, Martin Bäuml, and Rainer Stiefelhagen. "knock! knock! who is it?" probabilistic person identification in TV series. In *Proc. CVPR*, 2012. 2

[52] Makarand Tapaswi, Martin Bauml, and Rainer Stiefelhagen. Book2movie: Aligning video scenes with book chapters. In *Proc. CVPR*, pages 1827–1835, 2015. 2

[53] Makarand Tapaswi, Marc T. Law, and Sanja Fidler. Video face clustering with unknown number of clusters. In *Proc. ICCV*, 2019. 1, 2

[54] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *Proc. CVPR*, pages 4566–4575, 2015. 7

[55] Jingwen Wang, Wenhao Jiang, Lin Ma, Wei Liu, and Yong Xu. Bidirectional attentive fusion with context gating for dense video captioning. In *Proc. CVPR*, 2018. 2

[56] Teng Wang, Ruimao Zhang, Zhichao Lu, Feng Zheng, Ran Cheng, and Ping Luo. End-to-end dense video captioning with parallel decoding. In *Proc. ICCV*, 2021. 2

[57] Teng Wang, Huicheng Zheng, Mingjing Yu, Qian Tian, and Haifeng Hu. Event-centric hierarchical representation for dense video captioning. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020. 2

[58] Yujia Wang, Wei Liang, Haikun Huang, Yongqi Zhang, Dingzeyu Li, and Lap-Fai Yu. Toward automatic audio description generation for accessible videos. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2021. 2

[59] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman. Utterance-level aggregation for speaker recognition in the wild. In *Proc. ICASSP*, 2019. 4

[60] Yu Xiong, Qingqiu Huang, Lingfeng Guo, Hang Zhou, Bolei Zhou, and Dahua Lin. A graph-based framework to bridge movies and synopses. In *Proc. ICCV*, pages 4592–4601, 2019. 2

[61] Antoine Yang, Arsha Nagrani, Paul Hongsuck Seo, Antoine Miech, Jordi Pont-Tuset, Ivan Laptev, Josef Sivic, and Cordelia Schmid. Vid2seq: Large-scale pretraining of a visual language model for dense video captioning. *arXiv preprint arXiv:2302.14115*, 2023. 2, 9

[62] Youngjae Yu, Jiwan Chung, Heeseung Yun, Jongseok Kim, and Gunhee Kim. Transitional adaptation of pretrained models for visual storytelling. In *Proc. CVPR*, pages 12658–12668, 2021. 1, 2

[63] Youngjae Yu, Jongseok Kim, Heeseung Yun, Jiwan Chung, and Gunhee Kim. Character grounding and re-identification in story of videos and text descriptions. In *Proc. ECCV*, pages 543–559. Springer, 2020. 2

[64] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *Proc. ICLR*, 2020. 7

[65] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI*, 2018. 2

[66] Luowei Zhou, Yingbo Zhou, Jason J Corso, Richard Socher, and Caiming Xiong. End-to-end dense video captioning with masked transformer. In *Proc. CVPR*, 2018. 2

[67] Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proc. ICCV*, 2015. 2