Text-Conditioned Resampler For Long Form Video Understanding

Bruno Korbar[1,2]*, Yongqin Xian[2], Alessio Tonioni[2], Andrew Zisserman[1,3], and Federico Tombari[2,4]

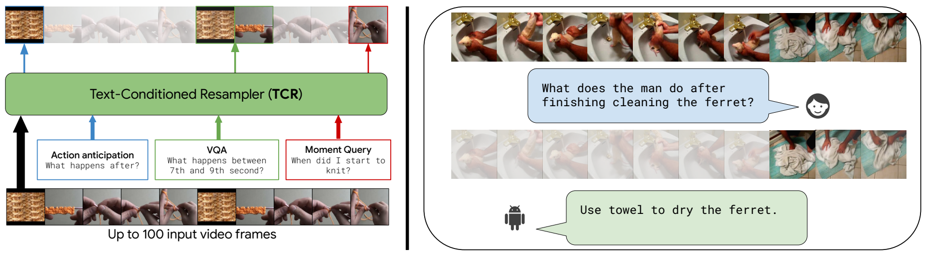
[1] Visual Geometry Group, University of Oxford
[2] Google, Zurich
[3] Google DeepMind, London
[4] TU Munich, Munich
*korbar@robots.ox.ac.uk

Abstract. In this paper we present a text-conditioned video resampler (TCR) module that uses a pre-trained and frozen visual encoder and large language model (LLM) to process long video sequences for a task. TCR localises relevant visual features from the video given a text condition and provides them to a LLM to generate a text response. Due to its lightweight design and use of cross-attention, TCR can process more than 100 frames at a time with plain attention and without optimised implementations. We make the following contributions: (i) we design a transformer-based sampling architecture that can process long videos conditioned on a task, together with a training method that enables it to bridge pre-trained visual and language models; (ii) we identify tasks that could benefit from longer video perception; and (iii) we empirically validate its efficacy on a wide variety of evaluation tasks including NextQA, EgoSchema, and the EGO4D-LTA challenge.

Keywords: video-understanding · visual-language models

1 Introduction

The development of visual-language models (VLMs) advanced exponentially in the past few years: new models pre-trained with increasingly larger scale, in terms of the number of parameters and size of the training set, continue pushing forward the state of the art on multiple tasks every couple of months. These models often have the ability to reason about the relationships of objects in their environment through natural language, often in an interactive fashion. This capability is appealing for multiple video applications. For example, it would be helpful for a model to be able to answer questions about a video: "Does this recipe use eggs?", "what does he do after he removes the tire?", etc. It is also appealing for users of augmented-reality devices: for example to be able to answer "where did I leave my phone?". Unfortunately, the computational requirements of such models made them impractical for use in video applications as the memory


Text-Conditioned Resampler (**TCR**)
Action anticipation
What happens after?
VQA
What happens between 7th and 9th second?
Moment Query
When did I start to knit?
Up to 100 input video frames

```
What does the man do after
finishing cleaning the ferret?
```

```
Use towel to dry the ferret.
```

Fig. 1: TCR resamples visual features that are relevant for the downstream tasks before passing them to the LLM. A qualitative example can be seen on the right.

requirement rises quadratically with the input size. Furthermore, to our knowledge, a large-enough source of even loosely labelled video data for training such a model from scratch does not readily exist.

That is why we are specifically interested in a subset of these models that are not trained from scratch, but rather 'bridge' pre-trained models via different types of 'visual-to-language adapter modules' [1, 25, 35]. The advantages of this approach, as opposed to training the model from scratch, are numerous: Only a small number of parameters are trained, which makes the memory footprint smaller; it allows us to utilise the capabilities of large visual backbones without overfitting to the downstream task; as well as to leverage the vast amount of knowledge stored in the LLM without suffering common limitations of smaller-scale fine-tuning such as catastrophic forgetting. Only a few of these models are trained on videos [1, 24, 54], and these can usually ingest only a small number of frames – typically anywhere between 4 to 32. Allowing a large number of video frames to interact with text is demonstrably beneficial [29, 35] in visual models, thus, a relatively simple way of increasing the model performance is to increase the number of frames the model sees.

In this paper we present a *Text-Conditioned Resampler* (TCR), an architecture and pre-training method that tackles all of the challenges mentioned above: it is a reasonably lightweight, low-dimensional adapter which acts as an information bottleneck between visual and language models. As shown in Figure 1 (left), it is able to process over a 100 (and up to 180) frames at a time, selecting the most relevant frame features to pass to the LLM based on the "conditioning" text. TCR allows us to focus on analysing videos with longer temporal span, and identify gains that could be made on longer videos. Right side of Figure 1 illustrates an application of our model. This new method allows us to analyse aspects of video datasets we've never been able to before. Specifically, we were able to look at how many frames it takes for a VLM to solve a task, and to determine if increasing the temporal span of perceived video actually brings benefits in terms of performance. We found that increasing temporal span does improve results on the moment-queries EGO4D challenge, and allows us to set the state-of-the-art (SOTA) on long-video question answering on the validation sets of EgoSchema dataset [29] and EGO4D long-term forecasting challenges, as well as on the temporally-sensitive NextQA dataset [48].

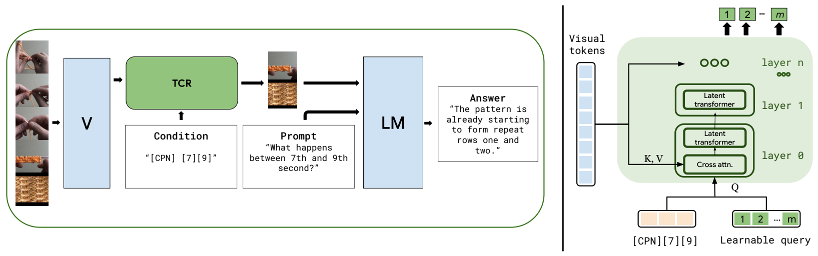


Fig. 2: Left: overview of how TCR integrates in a VLM in order to process long videos. A long (30-120 frames) sequence from a visual encoder (V) is resampled to a fixed-length sequence fed to a language model. `[CPN]` indicates special token for captioning; `[7][9]` is a representation of tokenised time steps. Right: details of the TCR module. Elements in blue are kept frozen. Best viewed in colour.

2 Text-Conditioned Resampler (TCR)

In the following section we describe the model and the training procedures used for training a video-specific VLM able to handle very long video sequences.

2.1 Model

At a high level, the input to the TCR consists of video frames processed by a visual encoder and embedded text tokens. It outputs a fixed-length sequence of embeddings that, together with a text prompt, is consumed by a language model. The text specifies (conditions) the task, and the TCR selects different visual features according to the task and transforms them to be suitable for input to the language model. Finally, the language model generates the text response to the specified task. Architecture overview is given in Figure 2 on the left.

Overview: The visual inputs consist of RGB frames of the video that are ingested by a pre-trained frozen ViT-g [38] model to obtain visual embeddings. Temporal encodings are added to them. The conditioning text tokens are prefixed with a learnable special token specifying the task the model is trying to solve and concatenated with a set of learnable query vectors. The queries and text interact with each other through self-attention layers, and interact with the frozen visual features through cross-attention layers (inserted every other transformer block). Output query vectors are then concatenated with an optional text prompt, and passed through a frozen Flan-T5 language model [9]. The TCR module is illustrated in Figure 2 on the right. We treat it as a plug-in replacement for the Q-former in the BLIP2 [25] architecture to enable handling of very long frame sequences as input.

The key design choices are: (i) the interaction of the query sequence with the visual features is only through cross-attention. This enables the TCR to ingest very long sequences (as it is not limited by the quadratic complexity of vanilla self-attention); and (ii) the output is a fixed length set (the transformed query vectors), so that the input to the language model is only a small number of tokens, irrespective of the length of the video sequence. Following these design principles we are able to significantly reduce the number of input tokens that the LLM needs to process with obvious gains in terms of inference time and memory requirements compared to full self-attention over all frame tokens.

How does the TCR differ from the Flamingo Resampler and Q-former? These design decisions build on the architectural innovations of the Perceiver resampler in Flamingo [1] and the Q-former in BLIP-2 [25]. However, there are a number of differences: (i) While Q-former is trained on images, TCR is optimised for video from the ground up – all training stages are done on videos. This is important as the TCR must learn to sample visual features from video frames conditioned on the task. (ii) TCR uses lower dimensional features than either Q-former or Perceiver Resampler (512 vs 768 vs 1536) and an overall smaller number of parameters (69M vs 188M). This is important as it allows us to process far longer video sequences. (iii) While TCR cross-attends visual features to text embeddings and learnable queries, Perceiver Resampler concatenates visual-embeddings and queries in a key-value pair, which makes the computation more expensive as it computes cross-attention and self-attention in a single pass. We keep the operations separate (i.e. first cross-attending text-query sequence with the video, and then self-attending the text-query sequence). This reduces per-layer computational requirements allowing us to increase video sequence length. These differences lead to a novel capability of processing many more frames at once, which subsequently leads to superior performance on downstream tasks.

Conditioning sequence construction: Most tasks can be represented as a basic Question and Answer (QA) pair. Inspired by multi-task language models [33], we adopt a generic `[ST][task prompt][learnable query]` input structure (where `[ST]` is a task-specific special token, `[task prompt]` is, for example, a question in a QA, and `[learnable queries]` are passed on to the LLM). We prefix a special task token (`[CPN]`, `[TRG]`, `[QA]`, `[STG]`) for captioning, temporal grounding, question-answering, and spatio-temporal grounding respectively) to the task prompt, depending on what task the model is solving. Figure 2 shows an example wih the `[CPN]` task-specific special token. Since, in principle, *all* tasks can be formulated as QA (and would be specified in the model as `[QA][question text]`), why are special tokens used? We found that using tokens improves overall performance while making the model easier to train and reducing the sequence length required for conditioning the sampler (as opposed to spelling out the task in text).

2.2 Training

Recent works have shown that contrastive learning yields visual representations for video frames that perform better in discriminative tasks than training in a purely generative fashion [24, 53]. Training models with a generative loss, however, seems to be crucial for developing reasoning regarding temporal grounding of unconstrained videos as well as the semantic relationship between text structure and video [1, 50]. Hence, we separate our training in three distinct stages:

Table 1: Effect of initialisation and pre-training stages on NextQA question answering and NLQ task. For NextQA, we use shortened fine-tuning procedure (see Section 3.2) and vary the checkpoints used. For NLQ, we evaluate on TCR w/LLM.

Init	Pre-training (i)	(ii)	(iii)	NextQA Acc ↑	NLQ MR@1 ↑
✓	✓	✓	✓	**66.1**	**11.42**
✓	✗	✗	✗	52.1	7.88
✗	✓	✓	✓	63.3	9.41
✓	✓	✗	✗	64.1	8.94
✓	✓	✓	✗	65.6	9.37
✓	✗	✓	✗	63.4	8.91
✓	✓	✗	✓	64.2	8.13

(i) initialisation, where we train TCR without the LLM; (ii) pre-training, where we train TCR in conjunction with the LLM; and later, (iii) a task-specific fine-tuning. Note that the only thing we're training is the TCR module – the visual encoder and LLM remain frozen throughout. Initialisation and pre-training stages are done on the YTT-1B dataset [50]. Videos in this dataset are annotated by the transcribed speech sentences and their corresponding timestamps that are either user-generated or automatically generated via automatic-speech recognition. Speech in such videos is rarely visually grounded [16, 22], however, because our model can see the video sequence surrounding the annotated segment, it is well suited to implicitly learn the temporal grounding. We describe training stages below.

Initialisation (without LLM): To initialise our model, we follow BLIP2 [25]'s *image-text contrastive* and *image-text matching* objectives. Contrastive objective maximises mutual information between TCR text output, and learnable queries which are cross-attended with a video. Text and learnable queries are passed together to TCR. Their mutual attentions masked in such a way that text only attends to itself, while the learnable queries are cross-attended to the video frames and then self-attended to themselves. We compute the average of text queries to get a text representation t, and compare it pairwise with all learnable queries. Query with maximum similarity to t is denoted as q. We then align the representations t and q by contrasting each positive pair with in-batch negative pairs. At this stage TCR is *not* text conditioned. Image-text matching objective (video-text matching in our case) primes the model for text-conditioning. Both learnable queries and text are passed through TCR together, without attention masking. A binary classifier predicting whether the video and text are matching or not is applied to each of the learnable queries and predictions are averaged to obtain a final matching score. The negatives are sampled in-batch, following [25].

We skip the *generative* training step of [25], as our model is neither designed nor initialised from a language model, and we found no measurable benefit from this training stage. The reader is referred to the original paper [25] for in-depth description of attention-masks and losses used during each of the objectives.

Pre-training (with LLM): The goal of pre-training is twofold: first, to semantically and temporally align TCR's output with the expected input of the LLM, and second to train TCR's self-attention layer to attend to specific task-specifying special tokens and text conditioning tokens. We do this by training it on three tasks. (i) given an untrimmed video and annotated sentence, we ask it to retrieve *when* the sentence occurred; (ii) given the untrimmed video and a timestep, we ask the model to fully caption that particular segment; (iii) given the untrimmed video and a text sequence corrupted in multiple ways, we ask it to correct the sequence. All tasks are supervised by applying the generative loss on the outputs of the LLM. The examples of these tasks on an example from YTT dataset can be seen in the supplementary material. The effects of these training stages can be seen in Table 1.

Fine-tuning: After these two stages, TCR achieves competitive results on downstream tasks while still being a generalist model. However, as our pre-training dataset is comprised mostly of low- and mid-quality videos with noisy automatic annotations, we observe significant improvements through fine-tuning for a specific task. The goal of fine-tuning is to align the TCR with the domain of the downstream task in question. Only the TCR module and its vocabulary are fine-tuned, while the visual encoder and the LLM are kept frozen. Fine-tuning is performed on each of the downstream datasets and is described in the results section for each dataset, while hyperparameters and ablation of the performance with or without fine-tuning are given in the supplementary.

2.3 Model details

Video sequence construction: We extract visual representations (14×14 patches from frames with 224^2 resolution) using ViT-g [38], and add temporal embeddings. In order to reduce memory consumption, for every other frame we drop random 50% of its patches. Recent work [17, 40] has shown no significant loss in performance when random patches have been dropped.

LLM sequence construction: We follow BLIP2 in the way we construct the input sequence [25]. We concatenate the output of the TCR module together with a `<BOS>` (beginning of sentence) token and the instruction context tokens (for example question in VQA, or previous action sequence together with instruction for EGO4D action prediction).

TCR architecture details: TCR is based on a transformer-decoder module [42], consisting of 4 transformer blocks with 8 attention heads and hidden dimension equal to 512. Blocks 0 and 2 contain cross-attention layers. For each task we use 128 512-dimensional queries. These choices were tuned based on the downstream performance on NextQA validation set and then kept fixed.

3 Experiments

In the following section, we conduct a set of experiments with a baseline VLM and TCR in order to determine which tasks benefit from having access to longer or denser video sequences, and compare the results to the SOTA. Specifically, we analyse the datasets in section 3.1. We compare the results to the state of the art in sections 3.2, 3.3 and 3.4, and we present ablation of model decisions in section 3.5. Qualitative results can be seen in Figure 4.

Datasets: We evaluate the following datasets: Kinetics400 [21] containing around 260k 10s videos with human-action labels and Countix, a subset of Kinetics where actions are annotated with the number of repeats (e.g. how many time a push up is repeated) [12]. MSR-VTT [49], a large scale video captioning dataset. NextQA, a manually annotated video-question-answering dataset where the model is asked to answer questions regarding temporal actions in a multiple-choice fashion [48] from (on average) 44s long videos. Finally, since egocentric videos are a new frontier in effective long-term video understanding, we evaluate on two diverse challenges from EGO4D [15]. EGO4D videos are often minutes long, containing both fine-grained actions as well as long-term interactions [15].

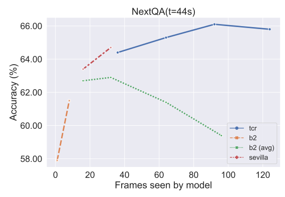
Baseline: We use a fixed BLIP2 [25] VLM as a baseline throughout our experiments. BLIP2 is not trained on videos, but it has been shown that it can be adapted to videos [53,57] and we follow [53] to do so. BLIP2 can only process up to 8 frames at a time, so for the task where it can be done, we average predictions over multiple 8-frame video clips extracted at 1fps (noted as 'BLIP2(Avg.)'). These aggregation methods, however, can introduce unwanted noise [23,35].

Modelling longer sequences: The design of TCR allows a VLM to "see" more frames than ever before. Therefore, we also present results using TCR which uses the same visual encoder and LLM as BLIP2 but is able to process *all* the frames at once. With TCR, each video can be processed in a single forward pass thus eliminating the effects of subsampling or averaging.

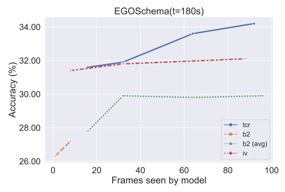
3.1 Video task analysis

Since videos are a highly redundant data-source, one has to ask how many frames, and at what sampling rate, does the model actually *need* to see to achieve good performance. For example, it has been observed that humans solve QA tasks with 8% higher accuracy when videos are sampled at 25fps as opposed to sampling them at 1fps [29]. In this section, we analyse the results on six common video-understanding tasks with respect to the number of frames consumed by the model. We look at the results from a high-level perspective, in order to determine which tasks will require higher number of frames for a model to solve. Overview of the results can be seen in Figure 3. BLIP2 and TCR models are initialised from the same checkpoint, and then finetuned on the downstream task using the target number of frames as an input. We describe evaluation procedure for each task in more detail in their respective sections.

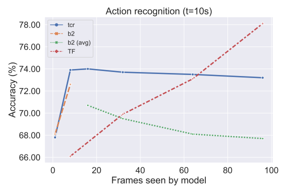
Intuitively, question-answering is one of the tasks where longer spans and better understanding of temporal dependencies would be of utmost importance. On the NextQA dataset, where questions contain temporal aspects (3a), seeing the span of an entire video seems to be crucial for performance. There is a clear peak at about 2fps, and a sharp decline past 1fps. This means that it is important to observe most of the frames, but frame density is not strictly required.



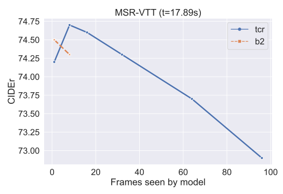
(a)



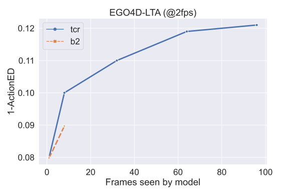
(b)



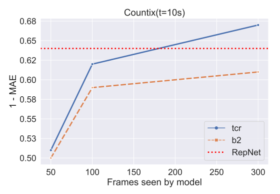
(c)



(d)



(e)



(f)

Fig. 3: Performance vs number of frames utilised by the models on various different tasks. t denotes average length of the video in the dataset. 'tcr'=Ours, 'iv'=IntenVideo [45], 'TF'=TimesFormer [2], 'b2'=BLIP2 [25], 'sevilla'= [54], RepNet= [12]

Curiously, although long video sequences at higher frame-density are required for humans to solve problems in EgoSchema dataset (3b), most models' performance actually peak or plateaus at significatly smaller number of frames [29]. We argue that this is because they have not been trained with long input length, and subsequently fail to capture semantic interdependencies within the video. TCR's performance increases with the number of frames, but plateaus when more sparsity in frame patches is needed to keep memory consumption down (see Table 3 for more details). We believe that being able to *see* the span of entire video with more density (i.e. at sampling rates greater than 1fps) could further increase performance on this benchmark.

Human action recognition (3c) and short-video captioning (3d) are commonly used video-understanding benchmarks, however, we found that they do not require many frames to achieve strong performance with a VLM. On action recognition, specialised models such as [2] scale better with higher frame density, however, which can be attributed to their to inherently learned sampling [23]. This intuition can be corroborated by a slight performance increase when using TCR module with BLIP2. Future prediction tasks (3e) on egocentric videos often span a long temporal range. We found that increasing the number of frames, hence covering larger video spans, helps in reducing the overall error. This is not unexpected, as action sequences in EGO4D tend to be repetitive, so the longer sequence of actions allows the model to recognise the action pattern from more samples. Finally, counting (3f) is an example of the task where frame density matters, and performance drops when fewer frames are utilised from a fixed length video. It also requires specialised architecture to solve it [12], and LLMs are traditionally disadvantaged in tasks that require numerical reasoning. For action classification and counting problems, we only use visual and aggregator parts of the VLM as described in the supplementary.

To conclude, although the tasks that require the models to be able to reason over many frames either in span or density are fairly limited, they do exist. Below, we show how a module such as TCR that allows us to 'see' more frames in context could be beneficial to overall performance on these tasks.

3.2 Evaluation on video question-answering

We first evaluate our model on question-answering benchmarks, our prior experiment shows a clear benefit when more frames are observed. We use the NextQA validation set to compare our model to the current state-of-the-art model which is also based on BLIP2 architecture [54]. We follow fine-tuning and evaluation protocol from [25], with hyperparameters outlined in the supplementary.

Input design: Video is subsampled to a target numbers of frames (92 frames at approximately 2fps for the final model), and temporal embeddings are computed accordingly. Conditioning text is formed as "`[QA] Question`" where `[QA]` is learned special token reserved for VQA tasks. During fine-tuning, the prompt to the LLM is formed following [54] as: "`[vis. features][question][options] Considering information in frames, select the correct answer`".

Evaluation procedure: During inference, we restrict generation to the answer vocabulary (i.e. "Option A", "Option B", ...), and select the most probable answer.

Comparison to SOTA: Results can be found in Table 2. Our model outperforms BLIP2 which demonstrates that the TCR module is successful in selecting the relevant frames, and also indicates the need for temporal modelling of this particular task. While like us, the SeViLA model is based on BLIP2, they train one model to sample relevant keyframes, and a separate model to solve the task from the sampled keyframes, effectively doubling the number of trainable parameters. In contrast, TCR requires only a single forward pass during training to both sample the features and solve the task. Our model outperforms SeViLA in overall accuracy (setting the new SOTA), hence showing that number of observed frames makes up for lack of trainable parameters.

Table 2: Comparison to SOTA on NextQA dataset. Results are split into non-balanced 'causal' (C), 'temporal' (T) and 'descriptive' (D) questions. The overall accuracy in the last column to the right is balanced across the entire dataset, rather than across the categories. '*' denotes re-implementation by [35]

Model	train params	accC ↑	accT ↑	accD ↑	acc ↑
SeViLA [54]	346M	73.4	68.8	**83.5**	73.4
HiTeA [51]	/	62.4	58.3	75.6	63.1
BLIP2 [25]	188M	64.9	59.7	77.8	63.5
2* [25, 35]	188M	72.9	65.2	80.1	70.1
Ours	76M	**73.5**	**69.8**	82.2	**73.5**

Table 3: Comparison to SOTA and human performance on EgoSchema split of EGO4D. × denotes multiple forward passes were used. * denotes higher proportion of patches was dropped.

Method	Observed frames	QA acc (%) ↑
InternVideo [7]	8×11	32.1
BLIP2 [25]	8	27.2
BLIP2 [25]	8×12	29.9
TCR (ours)	92	34.2
TCR (ours)	92×2	34.5
TCR (ours)	184*	**35.1**
Human	180	67.2

3.3 Evaluation on long-form VQA

EgoSchema is a long-form VQA dataset sampled from EGO4D containing 5000 human curated multiple choice question answer pairs, spanning over 250 hours of real video data. Each question requires the model to select one out of 5 possible answers and is accompanied by a three-minute-long video clip [29]. Input and evaluation designs are the same as they are for NextQA.
Comparison to SOTA: Results can be seen in Table 3. Our model outperforms both the SOTA models (where inference was done over multiple forward passes and prediction was averaged) and our re-implementation of BLIP2 (with both subsampled frames and iterative inference approach). Similar to [29], we observe relative saturation of performance with increasing the number of frames.

3.4 Evaluation on EGO4D challenges

Long-term action anticipation (LTA): The goal of the LTA challenge is to predict a sequence of twenty actions in order of appearance from an input video. The last observed action and action boundaries are given as well. The current state-of-the-art method relies solely on the power of large-language models in order to predict the sequence of future actions [18]. Our model adapts this idea but leverages the ability of TCR to process increasingly longer videos in order to achieve superior results. We compare our model to the SOTA, as well as to fine-tuned BLIP2 using 8 frames as video input. We note that our model outperforms BLIP2 by a significant margin, clearly showing the benefits of being able to observe denser video sequences for this task. Results can be seen in Table 4.
Input design: We construct input for fine-tuning and evaluation in the following fashion: video is subsampled uniformly to a target number of frames (8 for BLIP2 with Q-former, and 96 for BLIP2 with TCR) and temporal embeddings denoting the frame timestamp are added to them. The text prompts are designed as:

```
Complete an action sequence,
an action is one (verb, noun) pair.
A complete sequence consists of 28 actions.
Actions: (noun_1, verb_1) (verb_2, ...
```

and for the conditioning prompt we use:

```
[LTA][start_1](noun_1, verb_1),[start_2](noun_2, verb_2) ...
```

where `(noun, verb)` is an action pair, `[LTA]` is a learned special token, and `[start k]` is a tokenised start time of k-th action.
Evaluation procedure: Our evaluation procedure follows closely those of [18]. The model is fine-tuned to output comma-separated action pairs following the prompt formatting. During the evaluation, we softmax predictions over the reduced vocabulary of the label space for the LTA task. If both nouns and verbs fall into their respective label space, we append them to our prediction. For predictions with less than 20 action pairs, we pad it with the last action. Models denoted with (*) are sampled in an iterative fashion.
Comparison to SOTA: Table 4 shows the comparison to the state-of-the-art on long-term action prediction. Note that SOTA [60] uses various pre-processing methods in addition to the language model to predict the future actions. We use a single model. We find that iterative evaluation (i.e. asking the model to predict action by action, as opposed to the whole set of 20 actions is beneficial for the performance. Results can be improved by observing the (given) frames after the target time (See Tbl **X** in the supplementary).

Moment queries (MQ): The MQ task is similar to temporal action localisation or moment retrieval tasks. Given a textual description of an action, the goal is to localise all possible instances of it in the given video clip. Results can be seen in the Table 5.
Input design: Video is subsampled uniformly to the target number of frames, and temporal embeddings denoting the frame timestamps are added to it. The conditioning prompt is formed as "`[TRG] action query string`" where `[TRG]` indicates the special token for temporal grounding and `action query string` denotes the name of the action label parsed as a string. The language model is prompted by the following string

```
Return a sequence of frame timestamps
where <action name> is happening. The
timestamps range from 1 to 1000.
```

Table 4: Comparison of various models on the validation set of *EGO4D LTA challenge* (v2). Edit distance is reported and the lower the score the better. Models denoted with '*' are sampled iteratively. We use official implementation of [60] on v2 split.

Method	VerbED ↓	NounED ↓	ActionED ↓
PALM* [18]	0.7165	0.6767	0.8934
AntGPT* [60]	0.7083	0.6895	0.8826
BLIP2 [25]	0.7512	0.6873	0.9103
Ours	0.7009	0.6472	0.8792
Ours*	**0.6982**	**0.6441**	**0.8704**

Table 5: Comparison to the state of the art on the validation set of *Ego4D Moment Query Challenge*.

Method	Avg. mAP ↑	R@1, tIoU=0.5 ↑
Intern Video [7]	23.59	41.13
ASL [36]	**27.85**	**46.98**
Ours (96f)	24.51	42.99
Ours (192f)	25.45	43.72

Evaluation procedure: We softmax the model predictions from a reduced vocabulary of integers from 1 to 1000 (temporal coordinates are quantised similarly to [8]) and aggregate them.
Comparison to SOTA: Results in Table 5 show that despite the disadvantage of solving a discriminative task in a generative way, our model still performs admirably (-2.4 MAP) when compared to the state-of-the-art. In the supplementary material, we present an additional evaluation procedure (using direct classification) which can yield even better performance (+1.25 MAP).

3.5 Model design decisions

In the following section we investigate our model choices and seek to explain their impact on the performance of our model. All experiments were done on the validation set of NextQA dataset and results can be seen in Table 6. Note that we fine-tune the model on a shorter training schedule which yields lower results, but allows for a quicker turnaround. We keep the same fine-tuning parameters for all ablation studies.
Does text conditioning impact the results? We investigate the performance of our model in three different scenarios: (1) when the conditioning prompt is unchanged in the evaluation setting, (2) we completely remove the conditioning prompt, and (3) we modify the temporal word ('before' to 'after' and vice-versa) in a hope to confuse the model. The results can be seen in Table 6a. Conditioning indeed allows the TCR module to extract more relevant features (+3.8). Furthermore, adversarial conditioning greatly impacts the performance of the model (-7.6).

Table 6: Ablation studies on validation set of the NextQA dataset. Note that the ablations were done on a short training schedule.

cond.	acc ↑
yes	**64.9**
none	61.1
corrupt	55.3

(a) Different conditioning prompts on *temporal-question set only.*

#frms	acc ↑
32	64.4
92	**66.2**
124	65.9

(b) Impact of number of frames on model performance.

#queries	acc ↑
32	62.7
64	65.8
128	**66.2**
256	64.3

(c) Impact of the total number of queries on model performance.

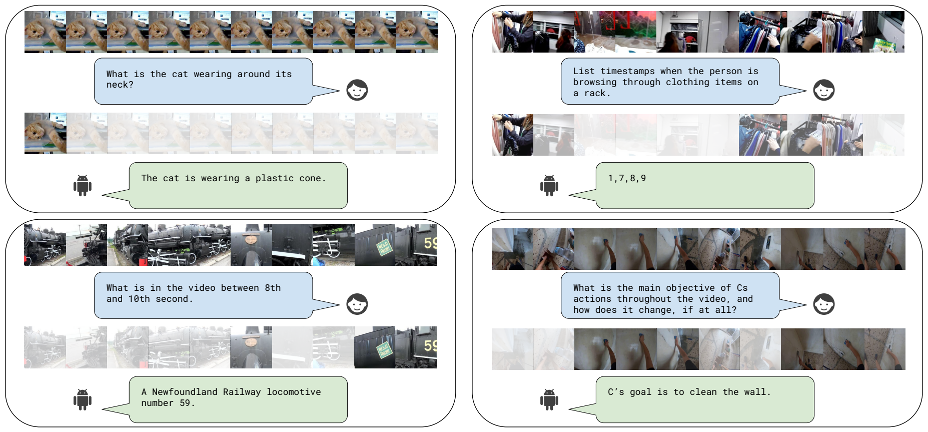

What is the cat wearing around its neck?
The cat is wearing a plastic cone.
List timestamps when the person is browsing through clothing items on a rack.
1,7,8,9
What is in the video between 8th and 10th second.
A Newfoundland Railway locomotive number 59.
What is the main objective of Cs actions throughout the video, and how does it change, if at all?
C's goal is to clean the wall.

Fig. 4: Examples of our model responding to various textual prompts taken from NextQA, EGO4D-MR, and YTT datasets. The opacity of the images in the second row is correlated to the mean patch attention score for that frame. Note that frames are subsampled and the TCR conditioning is not included for clarity.

Do we need special tokens for conditioning? If the model is fine-tuned for a specific task without the special tokens, it still performs reasonably well (73.5% vs 72.7% acc on NextQA with and without special tokens respectively).
Does the number of frames matter? The input video-sequence length is important to the model performance. In Table 6b we show the performance dependence on the input sequence length. Note that videos are on average 44s long, thus 124 frames equates to sampling at a rate of 2.5fps.
How many queries should the LLM see? While there is a benefit of perceiving a longer length of a video input-sequence, it has been observed that including more visual tokens as input to the LLM does not lead to a better performance [54]. Therefore in Table 6c we investigate how many queries the LLM should observe. Reducing the number of queries to a total of 128 (equivalent to four frames according to [25]) achieves optimal performance.

4 Related work

Our work spans many fields of video-understanding and we outline the most relevant related work below.
Video-sampling techniques: Sampling relevant frames from videos has long been a challenge in video understanding due to the highly redundant nature of video data. These methods either use a pre-processing module [3,6,14,23,46,52,61] to guide their model through multi-modal attention-like mechanisms [13,31], or employ recursive reinforcement learning techniques [47] to select relevant parts of the videos. In temporal action detection, models are tasked with precisely locating action boundaries. Most commonly used datasets [4,44,59] are dominated by custom solutions or transformer architectures [36,56] built upon strong features [5,40,41,43,45].

Egocentric videos understanding: Extreme length and temporally sensitive nature of the tasks introduced in EGO4D required researchers to think about the problem of video-length on a different scale [15,30]. This has already yielded creative approaches to solve various challenges in the egocentric space [7,20,39]. Also new and exciting benchmarks have been developed: for example the recent EgoSchema dataset [29], a manually annotated subset of EGO4D where each QA pair corresponds to a 3 minute-long video. A work particularly relevant to ours is SpotEM [34], a lightweight sampling mechanism that makes use of low dimensional features in order to select video segments important for solving natural-language query challenge . Though impressive in performance, their method is limited to the type of embeddings used for sampling, and is thus less general than our approach.

Video-language models and feature resampling: VLMs have revolutionised the field of computer vision – the scale of the models and data they were trained on increased exponentially in a short period of time [1,19,25,32,55], some even being jointly optimised for images and video [1,10,24,25]. The length of the videos these models can process often varies – [1] can process up to 8 frames, [24] can process longer tubelets (at reduced receptive fields). None of these models can process videos over 16 frames at full resolution outright.

Concurrent works on VLMs for video: Extending capabilities of VLMs is a fast-paced area of research, and many works have appeared without being published. [26] conducted an orthogonal study, exploring how the embedding quality can reduce the amount of visual information necessary for large LLMs. We on the other hand introduce a bottleneck module to increase the amount of data processed without increasing complexity. [28] focuses on interactive aspects by improving the LLM pipeline. We keep the pipeline fixed to seek improvements from the data. [58] increases the amount of information by introducing additional modalities which would be an interesting next step for our work as well. Similar to us, [37] aims to increase video length in a BLIP2 model, but they do so via a memory bank. Combining a memory-augmented approach with reasoning over longer sequences would be a promising future work. Techniques like FlashAttention [11] or RingAttention [27] also allow the context window of a VLM to handle long sequences of frames but at the cost of significant growth in inference speed. These techniques are complementary to our proposal and could be integrated in the TCR to support even longer videos in the future.

5 Conclusion

We present a parameter-efficient, text-conditioned module and training method for bridging video-to-text gap that can be applied to a large number of frames in videos. Even though our model is entirely based on BLIP2 [25] architecture, introducing it in other VLMs would be straightforward. We believe that models capable of perceiving long video sequences such as TCR will open up a promising new direction in research.

References

1. Alayrac, J.B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., Ring, R., Rutherford, E., Cabi, S., Han, T., Gong, Z., Samangooei, S., Monteiro, M., Menick, J., Borgeaud, S., Brock, A., Nematzadeh, A., Sharifzadeh, S., Binkowski, M., Barreira, R., Vinyals, O., Zisserman, A., Simonyan, K.: Flamingo: a Visual Language Model for Few-Shot Learning (2022), `http://arxiv.org/abs/2204.14198`
2. Bertasius, G., Wang, H., Torresani, L.: Is space-time attention all you need for video understanding? In: ICML. vol. 2, p. 4 (2021)
3. Buch, S., Eyzaguirre, C., Gaidon, A., Wu, J., Fei-Fei, L., Niebles, J.C.: Revisiting the" video" in video-language understanding. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2917–2927 (2022)
4. Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J.: Activitynet: A large-scale video benchmark for human activity understanding. In: Proceedings of the ieee conference on computer vision and pattern recognition. pp. 961–970 (2015)
5. Carreira, J., Zisserman, A.: Quo vadis, action recognition? a new model and the kinetics dataset. In: proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 6299–6308 (2017)
6. Chen, D., Bilgic, M., Getoor, L., Jacobs, D.: Dynamic processing allocation in video **33**(11), 2174–2187 (2011)
7. Chen, G., Xing, S., Chen, Z., Wang, Y., Li, K., Li, Y., Liu, Y., Wang, J., Zheng, Y.D., Huang, B., Zhao, Z., Pan, J., Huang, Y., Wang, Z., Yu, J., He, Y., Zhang, H., Lu, T., Wang, Y., Wang, L., Qiao, Y.: Internvideo-ego4d: A pack of champion solutions to ego4d challenges (2022)
8. Chen, T., Saxena, S., Li, L., Fleet, D.J., Hinton, G.: Pix2seq: A language modeling framework for object detection. arXiv preprint arXiv:2109.10852 (2021)
9. Chung, H.W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al.: Scaling instruction-finetuned language models. arXiv preprint arXiv:2210.11416 (2022)
10. Dai, W., Li, J., Li, D., Tiong, A.M.H., Zhao, J., Wang, W., Li, B., Fung, P., Hoi, S.: Instructblip: Towards general-purpose vision-language models with instruction tuning (2023)
11. Dao, T., Fu, D.Y., Ermon, S., Rudra, A., Ré, C.: Flashattention: Fast and memory-efficient exact attention with io-awareness (2022)
12. Dwibedi, D., Aytar, Y., Tompson, J., Sermanet, P., Zisserman, A.: Counting out time: Class agnostic video repetition counting in the wild. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10387–10396 (2020)
13. Gao, R., Oh, T.H., Grauman, K., Torresani, L.: Listen to look: Action recognition by previewing audio. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10457–10467 (2020)
14. Gowda, S.N., Rohrbach, M., Sevilla-Lara, L.: Smart frame selection for action recognition. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 1451–1459 (2021)
15. Grauman, K., Westbury, A., Byrne, E., Chavis, Z., Furnari, A., Girdhar, R., Hamburger, J., Jiang, H., Liu, M., Liu, X., Martin, M., Nagarajan, T., Radosavovic, I., Ramakrishnan, S.K., Ryan, F., Sharma, J., Wray, M., Xu, M., Xu, E.Z., Zhao, C., Bansal, S., Batra, D., Cartillier, V., Crane, S., Do, T., Doulaty, M., Erapalli, A., Feichtenhofer, C., Fragomeni, A., Fu, Q., Fuegen, C., Gebreselasie, A., Gonzalez,

C., Hillis, J., Huang, X., Huang, Y., Jia, W., Khoo, W., Kolar, J., Kottur, S., Kumar, A., Landini, F., Li, C., Li, Y., Li, Z., Mangalam, K., Modhugu, R., Munro, J., Murrell, T., Nishiyasu, T., Price, W., Puentes, P.R., Ramazanova, M., Sari, L., Somasundaram, K., Southerland, A., Sugano, Y., Tao, R., Vo, M., Wang, Y., Wu, X., Yagi, T., Zhu, Y., Arbelaez, P., Crandall, D., Damen, D., Farinella, G.M., Ghanem, B., Ithapu, V.K., Jawahar, C.V., Joo, H., Kitani, K., Li, H., Newcombe, R., Oliva, A., Park, H.S., Rehg, J.M., Sato, Y., Shi, J., Shou, M.Z., Torralba, A., Torresani, L., Yan, M., Malik, J.: Ego4d: Around the World in 3,000 Hours of Egocentric Video. In: IEEE/CVF Computer Vision and Pattern Recognition (CVPR) (2022)
16. Han, K., Rebuffi, S.A., Ehrhardt, S., Vedaldi, A., Zisserman, A.: Automatically discovering and learning new visual categories with ranking statistics. In: International Conference on Learning Representations (2020)
17. Han, T., Xie, W., Zisserman, A.: Turbo training with token dropout. In: BMVC (2022)
18. Huang, D., Hilliges, O., Gool, L.V., Wang, X.: Palm: Predicting actions through language models @ ego4d long-term action anticipation challenge 2023 (2023)
19. Jia, C., Yang, Y., Xia, Y., Chen, Y.T., Parekh, Z., Pham, H., Le, Q.V., Sung, Y., Li, Z., Duerig, T.: Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision. arXiv:2102.05918 [cs] (2021), `http://arxiv.org/abs/2102.05918`, arXiv: 2102.05918
20. Jiang, H., Ramakrishnan, S.K., Grauman, K.: Single-stage visual query localization in egocentric videos. arXiv preprint arXiv:2306.09324 (2023)
21. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., Zisserman, A.: The kinetics human action video dataset (2017)
22. Ko, D., Choi, J., Ko, J., Noh, S., On, K.W., Kim, E.S., Kim, H.J.: Video-text representation learning via differentiable weak temporal alignment. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (2022)
23. Korbar, B., Tran, D., Torresani, L.: Scsampler: Sampling salient clips from video for efficient action recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6232–6242 (2019)
24. Kuo, W., Piergiovanni, A.J., Kim, D., Luo, X., Caine, B., Li, W., Ogale, A., Zhou, L., Dai, A., Chen, Z., Cui, C., Angelova, A.: MaMMUT: A Simple Architecture for Joint Learning for MultiModal Tasks (2023), `http://arxiv.org/abs/2303.16839`
25. Li, J., Li, D., Savarese, S., Hoi, S.: BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models (2023), `http://arxiv.org/abs/2301.12597`
26. Li, Y., Wang, C., Jia, J.: Llama-vid: An image is worth 2 tokens in large language models (2023)
27. Liu, H., Zaharia, M., Abbeel, P.: Ring attention with blockwise transformers for near-infinite context (2023)
28. Maaz, M., Rasheed, H., Khan, S., Khan, F.S.: Video-chatgpt: Towards detailed video understanding via large vision and language models (2023)
29. Mangalam, K., Akshulakov, R., Malik, J.: Egoschema: A diagnostic benchmark for very long-form video language understanding. In: Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track (2023)
30. Mavroudi, E., Afouras, T., Torresani, L.: Learning to ground instructional articles in videos through narrations. arXiv preprint arXiv:2306.03802 (2023)

31. Panda, R., Chen, C.F.R., Fan, Q., Sun, X., Saenko, K., Oliva, A., Feris, R.: Adamml: Adaptive multi-modal learning for efficient video recognition. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 7576–7585 (2021)
32. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision (2021), `https://arxiv.org/abs/2103.00020`
33. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research **21**(1), 5485–5551 (2020)
34. Ramakrishnan, S.K., Al-Halah, Z., Grauman, K.: Spotem: Efficient video search for episodic memory (2023)
35. Sevilla-Lara, L., Zha, S., Yan, Z., Goswami, V., Feiszli, M., Torresani, L.: Only time can tell: Discovering temporal data for temporal modeling. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 535–544 (2021)
36. Shao, J., Wang, X., Quan, R., Yang, Y.: Action sensitivity learning for the ego4d episodic memory challenge 2023. arXiv preprint arXiv:2306.09172 (2023)
37. Song, E., Chai, W., Wang, G., Zhang, Y., Zhou, H., Wu, F., Chi, H., Guo, X., Ye, T., Zhang, Y., Lu, Y., Hwang, J.N., Wang, G.: Moviechat: From dense token to sparse memory for long video understanding (2023)
38. Sun, Q., Fang, Y., Wu, L., Wang, X., Cao, Y.: Eva-clip: Improved training techniques for clip at scale. arXiv preprint arXiv:2303.15389 (2023)
39. Tan, R., De Lange, M., Iuzzolino, M., Plummer, B.A., Saenko, K., Ridgeway, K., Torresani, L.: Multiscale video pretraining for long-term activity forecasting. arXiv preprint arXiv:2307.12854 (2023)
40. Tong, Z., Song, Y., Wang, J., Wang, L.: Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. Advances in neural information processing systems **35**, 10078–10093 (2022)
41. Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., Paluri, M.: A closer look at spatiotemporal convolutions for action recognition. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 6450–6459 (2018)
42. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
43. Wang, L., Huang, B., Zhao, Z., Tong, Z., He, Y., Wang, Y., Wang, Y., Qiao, Y.: Videomae v2: Scaling video masked autoencoders with dual masking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14549–14560 (2023)
44. Wang, L., Qiao, Y., Tang, X., et al.: Action recognition and detection by combining motion and appearance features. THUMOS14 Action Recognition Challenge **1**(2), 2 (2014)
45. Wang, Y., Li, K., Li, Y., He, Y., Huang, B., Zhao, Z., Zhang, H., Xu, J., Liu, Y., Wang, Z., et al.: Internvideo: General video foundation models via generative and discriminative learning. arXiv preprint arXiv:2212.03191 (2022)
46. Wang, Y., Chen, Z., Jiang, H., Song, S., Han, Y., Huang, G.: Adaptive focus for efficient video recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 16249–16258 (2021)

47. Wu, Z., Xiong, C., Jiang, Y.G., Davis, L.S.: Liteeval: A coarse-to-fine framework for resource efficient video recognition. Advances in Neural Information Processing Systems **32** (2019)
48. Xiao, J., Shang, X., Yao, A., Chua, T.S.: Next-qa: Next phase of question-answering to explaining temporal actions. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9777–9786 (2021)
49. Xu, J., Mei, T., Yao, T., Rui, Y.: Msr-vtt: A large video description dataset for bridging video and language. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) (June 2016), `https://www.microsoft.com/en-us/research/publication/msr-vtt-a-large-video-description-dataset-for-bridging-video-and-language/`
50. Yang, A., Nagrani, A., Seo, P.H., Miech, A., Pont-Tuset, J., Laptev, I., Sivic, J., Schmid, C.: Vid2seq: Large-scale pretraining of a visual language model for dense video captioning. In: CVPR (2023)
51. Ye, Q., Xu, G., Yan, M., Xu, H., Qian, Q., Zhang, J., Huang, F.: Hitea: Hierarchical temporal-aware video-language pre-training. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15405–15416 (2023)
52. Yeung, S., Russakovsky, O., Mori, G., Fei-Fei, L.: End-to-end learning of action detection from frame glimpses in videos. In: CVPR. pp. 2678–2687 (2016)
53. Yu, K.P.: VideoBLIP, `https://github.com/yukw777/VideoBLIP`
54. Yu, S., Cho, J., Yadav, P., Bansal, M.: Self-chained image-language model for video localization and question answering. arXiv preprint arXiv:2305.06988 (2023)
55. Zellers, R., Lu, X., Hessel, J., Yu, Y., Park, J.S., Cao, J., Farhadi, A., Choi, Y.: MERLOT: Multimodal Neural Script Knowledge Models. arXiv:2106.02636 [cs] (Oct 2021), `http://arxiv.org/abs/2106.02636`
56. Zhang, C.L., Wu, J., Li, Y.: Actionformer: Localizing moments of actions with transformers. In: European Conference on Computer Vision. pp. 492–510. Springer (2022)
57. Zhang, H., Li, X., Bing, L.: Video-llama: An instruction-tuned audio-visual language model for video understanding. arXiv preprint arXiv:2306.02858 (2023), `https://arxiv.org/abs/2306.02858`
58. Zhang, H., Li, X., Bing, L.: Video-llama: An instruction-tuned audio-visual language model for video understanding (2023)
59. Zhao, H., Torralba, A., Torresani, L., Yan, Z.: Hacs: Human action clips and segments dataset for recognition and temporal localization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 8668–8678 (2019)
60. Zhao, Q., Wang, S., Zhang, C., Fu, C., Do, M.Q., Agarwal, N., Lee, K., Sun, C.: Antgpt: Can large language models help long-term action anticipation from videos? ICLR (2024)
61. Zhi, Y., Tong, Z., Wang, L., Wu, G.: Mgsampler: An explainable sampling strategy for video action recognition. In: Proceedings of the IEEE/CVF International conference on Computer Vision. pp. 1513–1522 (2021)