Efficient Motion-Aware Video MLLM

Zijia Zhao[1,2], Yuqi Huo[3], Tongtian Yue[1,2], Longteng Guo[1,2] ,
Haoyu Lu[4] , Bingning Wang[3], Weipeng Chen[3], Jing Liu[1,2†]

[1]Institute of Automation, Chinese Academy of Sciences
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences
[3]Baichuan Inc. [4]Renmin University of China

Abstract

*Most current video MLLMs rely on uniform frame sampling and image-level encoders, resulting in inefficient data processing and limited motion awareness. To address these challenges, we introduce **EMA**, an **E**fficient **M**otion-**A**ware video MLLM that utilizes compressed video structures as inputs. We propose a motion-aware GOP (Group of Pictures) encoder that fuses spatial and motion information within a GOP unit in the compressed video stream, generating compact, informative visual tokens. By integrating fewer but denser RGB frames with more but sparser motion vectors in this native slow-fast input architecture, our approach reduces redundancy and enhances motion representation. Additionally, we introduce MotionBench, a benchmark for evaluating motion understanding across four motion types: linear, curved, rotational, and contact-based. Experimental results show that **EMA** achieves state-of-the-art performance on both MotionBench and popular video question answering benchmarks, while reducing inference costs. Moreover, **EMA** demonstrates strong scalability, as evidenced by its competitive performance on long video understanding benchmarks.*

1. Introduction

Understanding video content is essential for comprehensive world modeling. Recent advancements in multimodal large language models (MLLMs) for video understanding leverage uniform frame sampling [12, 18, 22, 23, 28, 31, 38, 42, 44] and apply vision encoders [29, 41], which are originally designed for image processing, to analyze each video frame. This approach concatenates frame features temporally before inputting them into the LLM for complex analysis. Here, the vision encoder captures spatial semantics per frame, while the LLM primarily handles temporal comprehension. However, *does this framework genuinely achieve comprehensive video understanding?*

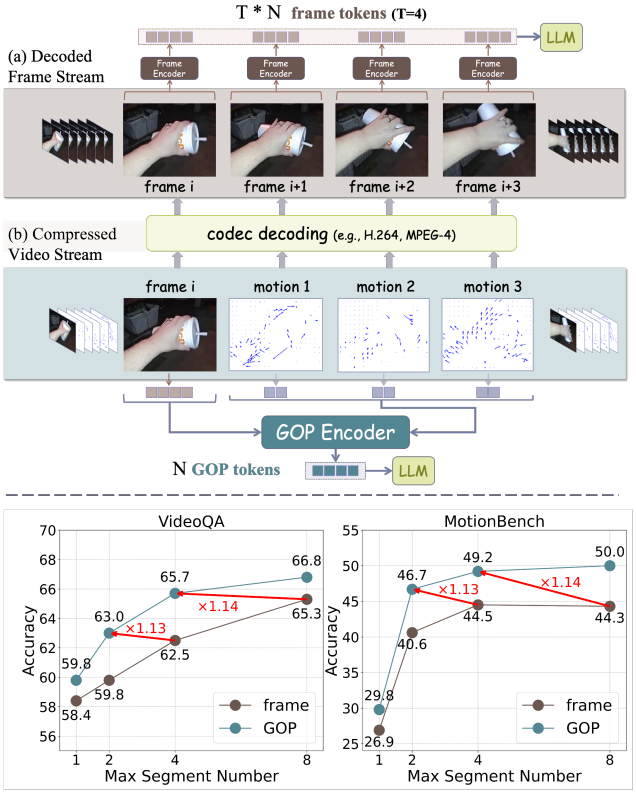

Figure 1. Comparison of sampling from decoded frames versus the GOPs (Group Of Pictures) from compressed video stream. Compressed video encoding generates tokens at only 1/T the length of sampled frames for the same clip, while capturing motion information more directly. It also shows greater efficiency on Video-QA (average of MSVD-QA and MSRVTT-QA [34]) and MotionBench within our **EMA** framework, achieving higher accuracy with less inference time (red arrow shows inference speed up).

The core difference between images and videos is that, unlike static images, videos contain motion information,

†Corresponding author.

which previous methods struggle to handle effectively. As illustrated in Fig. 1, consider a short segment from a video showing a person's hand holding a cup. When using uniform frame sampling, challenges arise: sampling a single frame (*e.g.*, frame i) may make it difficult, even for humans, to discern the cup's movement direction. To better capture motion, denser sampling is needed ((*e.g.*, frame $i \sim i + T$)), producing a longer sequence of visual tokens (*e.g.*, $T \times N$), which increases inference costs. Some methods [22, 36] improve efficiency in long video analysis by reducing the number of visual tokens per frame, often at the cost of frame-level semantic detail. Other approaches [5, 37, 45] exploit the redundancy between neighboring frames, employing a slow-fast encoding strategy to process large numbers of frames while compressing tokens through dynamic pooling. These methods mainly address the redundancy in individual frame features due to the similarity between adjacent frames in the video sequence. However, such redundancy can also be reduced prior to visual encoding.

We approach video encoding from a different perspective of video stream inputs. In conventional electronic video codecs (*e.g.*, MPEG-4, H.264), videos are stored with high compression ratios, significantly reducing redundancy. The compressed video stream holds enough information to fully reconstruct the video into frames, showing it contains what's needed to understand the video. Compression is achieved by segmenting video data into a limited set of RGB frames (I-frames) and a larger set of *sparse* motion vectors and residuals (P/B-frames). Most frames in a video sequence are reconstructed by referencing an I-frame and applying corresponding motion vectors and residuals. Using compressed videos [4, 9–11, 13, 17, 30] as input (with dense RGB values in I-frames and sparse motion vectors in P/B-frames) naturally forms a slow-fast input structure. This architecture minimizes redundancy compared to a fully decoded frame stream, lessening the need to compress redundant frames. Notably, the motion vectors inherently track the trajectories of macroblocks within each frame, making this compression-domain structure highly effective for capturing motion dynamics.

Building on this concept, we introduce an **E**fficient **M**otion-**A**ware video understanding MLLM **EMA**, which separately encodes key frames and motion frames from compressed video streams. Inspired by modern codecs' frame decoding, we propose a GOP encoder module to encode and integrate RGB and motion data, which segments video into GOP units and fuses spatial and motion semantics into the same number of visual tokens as a single frame while containing more information inside a video clip. The GOP feature sequence is then fed into a large language model for video understanding via a two-stage training. To evaluate motion understanding, we created MotionBench, a benchmark assessing four common types of motion: linear, curved, rotational, and contact-based. Our model achieved superior performance on both MotionBench and public benchmarks, including MSVD-QA [34], MSRVTT-QA [34], and ActivityNet-QA [40], while demonstrating reduced inference costs. Furthermore, when evaluated on long video understanding benchmarks, such as VideoMME [6], the model maintained competitive performance. These experimental results indicate that our approach not only lowers the inference cost of Video MLLMs but also enhances motion capture and overall video understanding.

Our contribution can be concluded as:

- We introduce **EMA**, an efficient motion-aware video model that leverages compressed video structures as slow-fast input to enhance motion capture and minimize input redundancy.
- Recognizing the lack of benchmarks dedicated to motion understanding, we develop MotionBench, a custom evaluation benchmark that assesses video motion understanding.
- Our model achieves competitive results on both MotionBench and public datasets with reduced inference costs. Additionally, it performs well on long video understanding benchmarks, demonstrating good context scalability.

2. Related Works

2.1. Video MLLMs

Recent developments in multimodal large language models have extended their applications to video data [31, 38, 42], where uniform frame sampling and frame-by-frame encoding are commonly employed. Early models, such as VideoChat [19], VideoChatGPT [28], and Valley [27], rely on mean pooling to aggregate frame-level features before feeding them to LLMs, which limits temporal comprehension. Furthermore, some advanced models have introduced specific enhancements to capture dynamic content; for instance, Chat-UniVi [12] utilizes dynamic token clustering, and BT-Adapter [25] incorporates lightweight adapters to improve image LLMs for video data. VideoChat2 [18] further replaces CLIP [29] with a dedicated video encoder [20], achieving better temporal modeling but at the cost of complex, multi-stage training. Video-LaVIT [13] decomposes video content into keyframes and motion discrete vectors, capturing temporal dynamics through a tokenization strategy.

By contrast, Our approach fundamentally departs from prior methods by directly leveraging compressed video data to efficiently capture both spatial and motion information in a unified architecture. Unlike methods that rely on separate tokenizer decomposition (*e.g.*, Video-LAVIT) or pooling strategies (*e.g.*, VideoChatGPT), our model processes video through a GOP encoder, simultaneously integrating keyframes and motion vectors within each GOP unit. This design reduces redundancy at the data input level, resulting in a more compact streamlined architecture that enhances

both efficiency and motion comprehension.

2.2. Compressed Video Understanding

The use of motion in video processing originates from early MPEG-4 codec-based research designed to enhance action recognition tasks [4, 9, 17]. Foundational methods like [17] employ a slow-i-fast-p architecture that prioritizes keyframes for spatial context while using sparse motion frames to efficiently capture temporal dynamics. Later, a self-supervised video representation learning approach [9] introduced context-motion decoupling for improved temporal understanding without explicit labels. Similarly, Mm-ViT [4] integrates multi-modal inputs, including motion data, within a video-transformer framework to improve action recognition. Subsequent works utilizing H.264 compression techniques (*e.g.*, [10, 11]) further optimized motion vector tokenization, demonstrating efficient motion data processing across extended video clips. Beyond action recognition, motion-aware frameworks have expanded into other video tasks. In the captioning domain, [30] employs motion vectors for real-time video descriptions, while VideoComposer [33] incorporates motion control for compositional video synthesis. Motion information has also been adapted for semantic segmentation [8] and prompt tuning [16].

Our work is a pioneering approach in integrating compressed video-based motion encoding within multimodal large language models for video understanding. By unifying motion and keyframe data in a GOP structure, our method captures spatiotemporal semantics with fewer tokens, achieving more efficient processing and marking a significant advancement over prior frame-sampled approaches.

3. Method

In this section, we mainly introduce the specific design of the proposed model **EMA**. At the same time, considering the current lack of motion capability detection for video MLLMs, we also manually construct a motion benchmark MotionBench.

3.1. Compressed-domain Video Input

We use compressed-domain videos as input to our video understanding model, which exhibit less redundancy compared to frame sampling inputs. In modern codecs (*e.g.*, H.264), video frames are categorized into three types: I-frames (intra-coded frames), P-frames (predictive-coded frames), and B-frames (bipredictive-coded frames). As shown in Fig. 2, I-frames are encoded independently and rely solely on the current frame, while P and B frames depend on adjacent reference frames , using these references along with the motion vectors and residuals stored at the current position to predict the current frame.

Motion Vector (MV) represents the displacement of a macroblock from the current frame to a reference frame. If

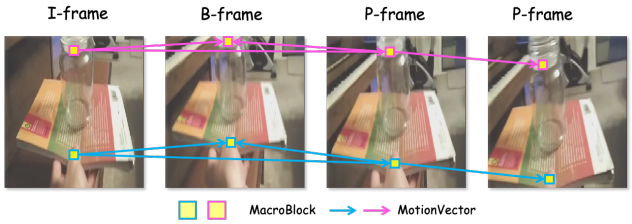

Figure 2. Description of the H.264 codec modes for a single GOP (Group of Pictures). P/B-frames are decoded sequentially in the decoding order. Decoding relies on motion vectors, which record the movement of each macroblock in the current frame relative to those in reference frames.

a macroblock at position (x, y) in the current frame corresponds to a macroblock at position (x', y') in the reference frame, the motion vector is defined as:

$$MV(x, y) = (x' - x, y' - y) \tag{1}$$

Residuals represent the prediction errors between the actual and predicted macroblocks. If $P(x, y)$ is the actual pixel value in the current frame, and $P'(x, y)$ is the predicted value from the reference frame with motion vector, the residual is:

$$\Delta(x, y) = P(x, y) - P'(x, y) \tag{2}$$

Thus, the decoded P/B frame is predicted using the reference frame I_{ref}, the motion vector MV, and the residual Δ:

$$I_{P/B} = \text{Pred}(I_{ref}, MV) + \Delta \tag{3}$$

Frames are grouped into Group of Pictures (GOP) units, each containing one I-frame and several P/B-frames, with each GOP decoding independently of others.

In our method, we segment the video input into several GOP units based on the codec structure of the source video. Within each GOP, we select the RGB values from the I-frame and uniformly sample a fixed number of forward motion vectors from the P/B-frames. This forms the compressed video segment input, defined as follows:

$$\begin{aligned}\text{Input}_{\text{Video}} = &\underbrace{[I_1, MV_{(1,1)}, MV_{(1,2)}, \ldots, MV_{(1,M)}]}_{\text{GOP}_1}, \ldots, \\ &\underbrace{[I_N, MV_{(N,1)}, MV_{(N,2)}, \ldots, MV_{(N,M)}]}_{\text{GOP}_N}\end{aligned} \tag{4}$$

Here, N represents the GOP segment number, M the motion vector frame number within a GOP, I_k the keyframe in the k-th GOP, and $MV_{(k,t)}$ the t-th motion vector frame in the k-th GOP. This GOP structure, compared to the decoded frame sequence, does not require additional decoding from the original data stream. Additionally, since motion vectors and residuals are more sparse than the decoded frames, this data format has less redundancy.

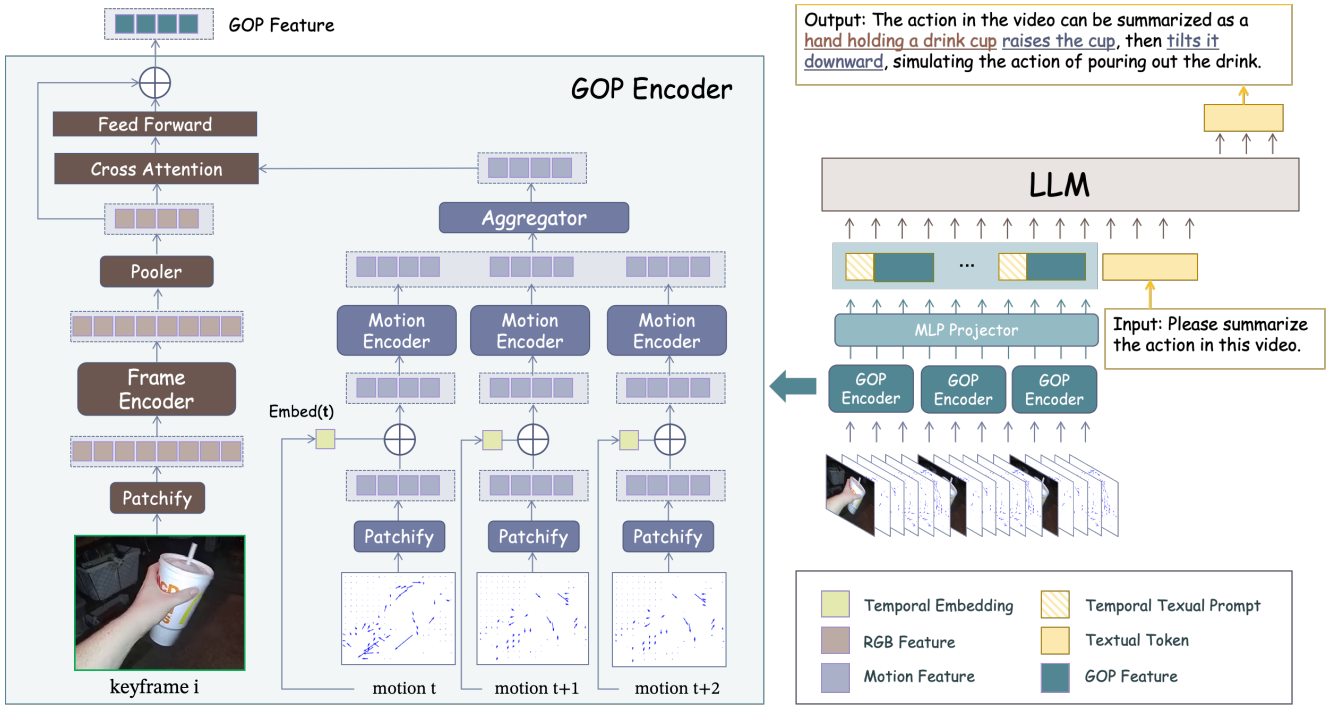

Figure 3. An illustrative diagram of the overall model architecture. The compressed-domain video stream is divided into GOPs (Groups of Pictures), and each GOP is encoded using our designed GOP encoder. After concatenation, the encoded GOPs are input into the LLM along with text instructions. On the left side of the figure is the detailed structure of the GOP encoder, which decouples frame and motion encoding. It fuses the frame features with the aggregated motion feature sequence to produce a fixed-length GOP feature containing both spatial and motion information.

3.2. Motion-aware GOP Encoder

We primarily encode the RGB and motion information within the GOP, utilizing the sparser compressed input for more efficient encoding compared to decoded frames. Our goal is to represent a video segment with the same number of tokens as a single frame, while capturing both spatial and motion information. To achieve this, we employ a decoupled encoding strategy, integrate motion position awareness, and apply attention fusion to build a motion-aware GOP encoder.

In each GOP, we decouple key frame and motion vector encoding using two separate encoders. A vision encoder, Enc_I, encodes the RGB values of the I-frame I_k, with dimensions $(h, w, 3)$. An adaptive pooling layer is then applied to reduce the number of visual tokens.

$$\boldsymbol{F}_k^I = \text{Pooling}\big(\text{Enc}_I(I_k)\big) \tag{5}$$

For motion vectors $MV_{(k,t)}$, with a macroblock size of 4, the dimensions are $(h//4, w//4, 2)$. Notably, learnable positional embeddings are added to the patchified motion vectors to preserve temporal information, as the reconstruction in the H.264 codec maintains temporal order.

$$\boldsymbol{F}_{(k,t)}^{MV} = \text{Enc}_{MV}\Big(\text{Patchify}(MV_{k,t}) + \text{PosEmbed}(t)\Big), \quad t = 1, \ldots, M \tag{6}$$

Then, we design a fusion module to fuse the key frame RGB feature $\boldsymbol{F}_k^I$ and motion feature sequence $\boldsymbol{F}_{(k,t),t=1,\ldots,M}^{MV}$ within an independent GOP segment $\text{GOP}_{\boldsymbol{k}}$. The resulting vision tokens in the GOP feature encapsulate both spatial semantics and motion information, while maintaining the same token count as a single frame feature. We aggregate the motion features using mean pooling along the temporal dimension.

$$\boldsymbol{F}_k^{MV} = \text{Aggregator}\Big(\big[\boldsymbol{F}_{(k,t)}^{MV}\big]_{t=1}^{M}\Big) \tag{7}$$

Then, we use a fusion layer to append motion feature $\boldsymbol{F}_k^{MV}$ on the spatial RGB feature $\boldsymbol{F}_k^I$. This layer includes a cross-attention mechanism, a feed-forward layer, and a residual bottleneck. The fusion layer aligns spatial objects with their corresponding motion semantics.

$$\boldsymbol{F}_k^{\text{attn}} = \text{Attention}\big(Q = \boldsymbol{F}_k^I, K, V = \boldsymbol{F}_k^{MV}\big) \tag{8}$$

$$\boldsymbol{F}_k^{\text{GOP}} = \text{FFN}\big(\boldsymbol{F}_k^{\text{attn}}\big) + \boldsymbol{F}_k^I \tag{9}$$

3.3. Overall Architecture

We divide the input videos into several GOP segments in compressed domain and encode each GOP into a fixed length of GOP features with the GOP encoder. An MLP modality projector then maps these GOP features to visual tokens.

These visual tokens are then concatenated in temporal order, with a temporal prompt TP indicating the time sequence of each GOP.

$$\boldsymbol{X}_V = \left[TP_1, \text{MLP}(\boldsymbol{F}_1^{\text{GOP}}), \ldots, TP_N, \text{MLP}(\boldsymbol{F}_N^{\text{GOP}})\right] \quad (10)$$

where TP is a tokenized textual phrase indicating the temporal information of current video segment $\text{GOP}_{\boldsymbol{k}}$, such as "Segment k". These visual tokens, along with human-provided textual instructions, are fed into a LLM to perform advanced video understanding tasks.

3.4. Training Strategy

We adopt a two-stage training strategy to training our model: visual-text alignment and visual instruction tuning.

Stage 1: Visual-Text Alignment. In this stage, we aim to build modality alignment between visual context and its textual description. We use image-text dataset LLaVA-558k [24] and video-text dataset Valley-702k [27] as training datasets. For images, we treat them as a static video represented by a single GOP, with a blank motion vector indicating no motion. The image or video is input into the model, and the model is tasked with generating the corresponding caption. The loss function for stage 1 is defined as follows:

$$\text{Loss}_{\text{stage1}} = -\sum_{\substack{i=1 \\ i \in \boldsymbol{X}_{\text{cap}}}}^{L} \log P_{\theta^*}(x_i \mid \boldsymbol{X}_V, \boldsymbol{X}_{\text{cap},<i}) \quad (11)$$

where L denotes the sequence length, $\boldsymbol{X}_{\text{cap},<i}$ represents the caption tokens preceding the i-th token, and θ^* refers to the trainable parameters in the motion encoder, GOP fusion layer, and modality projector.

Stage 2: Visual Instruction Tuning. In this stage, we train the model with visual instructions and responses to better understand human instructions in visual contexts. We collected instruction tuning data from image-text instruction dataset Cauldron [15] and video instruction dataset VideoChat2-IT [21], resulting in a total of 2.4M samples. The loss function for stage 2 is defined as follows:

$$\text{Loss}_{\text{stage2}} = -\sum_{\substack{i=1 \\ i \in \boldsymbol{X}_{\text{ans}}}}^{L} \log P_{\theta}\left(x_i \mid \boldsymbol{X}_V, \boldsymbol{X}_{\text{ins},<i}, \boldsymbol{X}_{\text{ans},<i}\right) \quad (12)$$

where L is the sequence length, $\boldsymbol{X}_{\text{ans},<i}$ and $\boldsymbol{X}_{\text{ins},<i}$ represent the tokens from the answer and instruction sequences preceding the i-th token, and θ denotes model parameters.

Motion Encoder Warmup. Since the motion encoder is a randomly-initialized transformer, we apply a warm-up phase to its parameters to facilitate faster convergence. To achieve this, we use Something-to-Something V2 (SSV2) dataset [7] and employ a supervised learning approach. Only motion vectors extracted from compressed videos are used as input, without any RGB information. An external classification head is added to the motion encoder, and it is trained with cross-entropy loss against the ground-truth action labels prior to Stage 1.

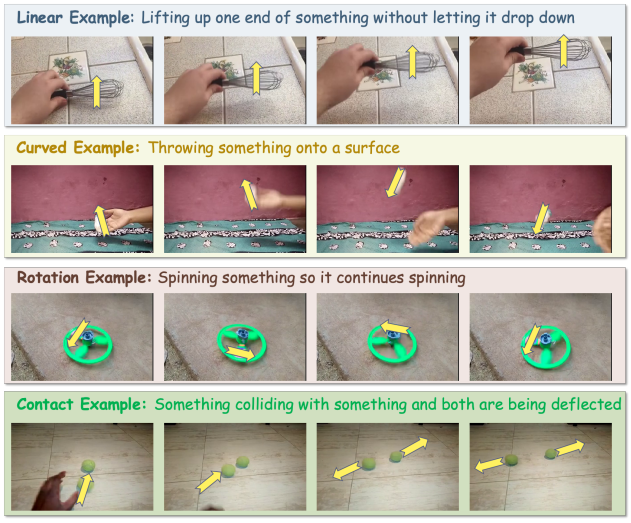

Figure 4. Examples of data from MotionBench. We show four types of examples: linear, curved, rotation, and contact. Yellow arrows in the video frames indicate the motion trajectories of the same specified object. Different trajectory patterns correspond to different data types.

3.5. MotionBench

We observed that existing video benchmarks lack specialized metrics for evaluating motion information, which hinders precise assessment of a model's sensitivity to motion dynamics. To address this gap, we introduce a custom benchmark, MotionBench, designed to evaluate video models' understanding of motion patterns. We collect 2.3k videos from various sources [7, 14, 32]. Each video in MotionBench contains a single distinct motion pattern without any scene transitions. Specifically, we collected 114 unique motion patterns, categorizing each video under one of these patterns manually as shown in Sec. 3.4. Based on the characteristics of motion trajectories, we classified the test data into four main categories: Linear (Lin.), Curved (Cur.), Rotation (Rot.), and Contact (Con.). The characteristics of each category are detailed below:

- **Linear**: Motion occurring in a consistent direction, such as a toy car moving from one side to another.
- **Curved**: Motion following a curved trajectory, like a ball thrown from a height toward the ground.
- **Rotation**: Clockwise rotational motion, such as a spinning CD or plate.
- **Contact**: Motion involving two objects making contact, often with compression, such as the collision of two balls.

During testing, we employed a multiple-choice format for each video, presenting one correct option alongside three carefully crafted incorrect options. These incorrect options

were intentionally selected as the most confusable motion patterns relative to the ground truth among the 114 classes, ensuring a rigorous evaluation of the model's ability to accurately comprehend motion dynamics. We report the multi-choice QA accuracy as the performance on MotionBench.

4. Experiments

4.1. Implementation Details

We employ Qwen2-7B [39] as the LLM backbone. The frame encoder is derived from SigLIP-so400m [41]. The motion encoder consists of a two-layer transformer with two channels. Details are provided in the Appendix.

We re-encode the videos using the H.264 codec with a maximum keyframe interval of 8 and a frame rate of 4 fps. During the training process, we uniformly sample groups of pictures (GOPs) from each video, limiting the maximum number of GOPs to 8. Each GOP consists of one RGB I-frame and a uniformly sampled set of motion vectors, with a maximum of 8 motion vectors. For image data, we treat each image as a static video segment; thus, an image GOP includes the RGB image and a zero motion vector. All RGB frames in a GOP are resized to 384×384 pixels, while all motion vectors are resized to 96×96 pixels, corresponding to a macroblock size of 4 in H.264 encoding. To enhance training and inference efficiency, the features of the RGB frames are processed using adaptive pooling with a 3×3 kernel. Each batch contains a random combination of images and videos, with data samples packed in a sequence limited to a maximum of 4096 tokens.

In the first stage, we train for 1000 steps with a batch size of 128, where only the motion encoder, GOP fusion layer, and modality projector are trainable. In the second stage, we train for 2400 steps with the same batch size, but all model parameters are fine-tuned. Our experiments are conducted on 16 NVIDIA A100 GPUs, and the training hyperparameters are detailed in the Appendix.

4.2. Quantitative Evaluation

4.2.1 Performance on Video-QA Benchmarks

We conduct a quantitative evaluation on public video question-answering benchmarks, including MSVD-QA [34], MSRVTT-QA [34], and ActivityNet-QA [40]. For this evaluation, we follow a standard approach [28], utilizing GPT-3.5-turbo to evaluate answers, providing both accuracy and score metrics. **EMA** outperforms various models using uniform frame sampling methods, such as Video-LLaVA [23], LLaMA-VID [22], and Chat-UniVi [12]. Video-LaVIT [13] also leverages an MPEG-4 codec to decouple keyframes and motion vectors. However, it requires a higher number of input tokens compared to **EMA**, as it simply concatenates keyframe and motion tokens without integrating them through fusion. **EMA** achieves superior performance over Video-LaVIT on MSVD-QA and ActivityNet-QA and achieves comparable results on MSRVTT-QA.

4.2.2 Extension on Long Video Benchmarks

Our approach can also be adapted for long video understanding tasks. We increased the model's maximum GOP count to 32 and trained it with the same setting. We evaluate on long video benchmark VideoMME [6]. Our results demonstrate that the GOP method is effective for longer video tasks, achieving competitive performance with most video models.

4.3. Efficiency Analysis

To evaluate the efficiency of our method, we report task performance, maximum input token count, and average inference time per sample in Tab. 3. The inference time refers to the average MLLM generation time per sample.

EMA achieves superior performance on MSVD-QA and MotionBench compared to previous methods such as Video-LLaVA [23] and LLaMA-VID [22]. Additionally, our model uses fewer visual tokens (648 vs. 2048) and demonstrates lower inference time (×3.08 compared to Video-LLaVA, ×2.14 compared to LLaMA-VID), highlighting its efficiency in video understanding.

We also compare model performance and inference time across different video segment types in Tab. 3. Specifically, we compare two input types: video frames, commonly used in previous models, and GOPs, used by **EMA**. All models are trained under identical settings, except for differences in segment type and count. Using GOPs as the video segment unit results in only a negligible increase in MLLM inference time compared to direct frame input, while maintaining the same visual token count. However, model-g achieve a significant performance boost over model-f using frame input with the same segment count, with a comparable inference time overhead. Notably, on MotionBench, GOPs outperform frames due to their superior ability to capture motion information. Additionally, we observe that GOP encoding outperforms models with denser frame sampling, as shown in Fig. 1 and Tab. 3, while also requiring less inference time (×1.13 comparing model-$g2/f4$ and ×1.14 comparing model-$g4/f8$). This demonstrates that our approach not only improves inference efficiency but also reduces redundancy in video inputs for video MLLMs.

4.4. Ablation Study

In this section, we conduct comprehensive ablation studies on module configuration within the GOP encoder. All models are trained on the same dataset and with identical hyperparameter settings, using a maximum of 4 GOPs.

Fusion module architecture in GOP encoder We evaluate different fusion strategies in the GOP encoder in Exp

Table 1. Comparison with SOTA models on public videoQA benchmarks: MSVD-QA [3]; MSRVTT-QA [35]; ActivityNet-QA [2]. Maximum values are in **bold**.

Method	MSVD-QA		MSRVTT-QA		ActivityNet-QA	
	Acc	Score	Acc	Score	Acc	Score
FrozenBiLM [38]	32.2	–	16.8	–	24.7	–
VideoLLaMA [42]	51.6	2.5	29.6	1.8	12.4	1.1
LLaMA-Adapter [44]	54.9	3.1	43.8	2.7	34.2	2.7
VideoChat [18]	56.3	2.8	45.0	2.5	26.5	2.2
Video-ChatGPT [28]	64.9	3.3	49.3	2.8	35.2	2.7
BT-Adapter [25]	67.5	3.7	57.0	3.2	45.7	3.2
LLaMA-VID [22]	69.7	3.7	57.7	3.2	47.4	3.3
Chat-UniVi [12]	65.0	3.6	54.6	3.1	45.8	3.2
Video-LLaVA [23]	70.7	3.9	59.2	**3.5**	45.3	3.3
MovieChat [31]	75.2	3.8	52.7	2.6	45.7	3.4
Video-LaVIT [13]	73.2	3.9	**59.3**	3.3	50.1	3.3
EMA	**75.8**	**4.1**	58.5	**3.5**	**52.1**	**3.5**

Table 2. Performance comparison with state-of-the-art models on long video tasks VideoMME [6]. Maximum values are in **bold**.

Method	VideoMME w/o subs				VideoMME w subs			
	short	medium	long	**Overall**	short	medium	long	**Overall**
Video-LLaVA [23]	45.3	38.0	36.2	39.9	46.1	40.7	38.1	41.6
Qwen-VL-Chat [1]	46.9	38.7	37.8	41.1	47.3	40.4	37.9	41.9
ST-LLM [26]	45.7	36.8	31.3	37.9	48.4	41.4	36.9	42.3
VideoChat2-Mistral [21]	48.3	37.0	33.2	39.5	52.8	39.4	39.2	43.8
Chat-UniVi-v1.5 [12]	45.7	40.3	35.8	40.6	51.2	44.6	41.8	45.9
LongVA [43]	61.1	50.4	46.2	52.6	61.6	53.6	47.6	54.3
EMA	65.8	50.2	44.3	**53.4**	65.3	55.4	54.3	**58.4**

Table 3. Efficiency analysis under different model, video segment type and number. We report the inference time and maximum visual token count for MLLM generation. We use model-f and model-g to denote models trained with frame and GOP inputs, respectively. Model-D refers to the final model used as **EMA**. Maximum values are in **bold**, and second-highest values are underlined.

Models	Segment Type	Max Seg. Number	MSVD Acc./Score	MSRVTT Acc./Score	MotionBench Lin.	Cur.	Rot.	Con.	Avg.	#Visual Tokens	Inference Time
Video-LLaVA [23]	frame	8	70.7 / 3.9	**59.3** / 3.5	36.1	31.2	32.0	44.5	36.0	2048	391.4ms
LLaMA-VID [22]	frame	8	69.7 / 3.7	57.7 / 3.2	34.1	35.8	37.0	43.0	37.5	2048	273.7ms
model-$f1$	frame	1	70.1 / 3.7	46.7 / 3.1	21.9	33.9	25.7	23.1	26.9	81	93.5ms
model-$g1$	GOP	1	71.2 / 3.7	48.4 / 3.2	26.3	35.5	30.0	27.4	29.8	81	94.0ms
model-$f2$	frame	2	71.0 / 3.7	48.6 / 3.2	42.6	39.1	44.3	34.5	40.6	162	97.0ms
model-$g2$	GOP	2	72.4 / 3.8	53.6 / 3.4	52.6	41.8	47.7	45.5	46.7	162	97.4ms
model-$f4$	frame	4	72.4 / 3.8	52.6 / 3.3	53.5	39.8	45.7	36.3	44.5	324	110.8ms
model-$g4$	GOP	4	74.3 / 4.0	57.0 / 3.5	59.8	42.7	47.7	46.6	49.2	324	111.5ms
model-$f8$	frame	8	74.0 / 4.0	56.7 / 3.4	53.5	40.4	43.3	37.1	44.3	648	127.1ms
model-$g8$ (**EMA**)	GOP	8	**75.8** / 4.1	58.5 / 3.5	60.1	43.6	49.3	47.0	**50.0**	648	127.1ms

(0-4) of Tab. 4. The add method simply sums the motion features and frame features within a GOP. The concat method concatenates motion and frame features along the last dimension. The crossattn method utilizes a cross-attention module to integrate motion information into the frame embeddings. Additionally, we experiment with using the entire set of motion features as the key/value inputs in the cross-attention module without first aggregating them through a pooling layer. The results indicate that the crossattn method yields a slight improvement over the add and concat methods on Video-QA tasks. Notably, it provides greater gains on MotionBench, suggesting that our fusion module architecture effectively integrates motion information with the semantics of static frames. We also observe that a pooling aggregator

Table 4. Ablation experiments on the method design. We compared the construction of the fusion module, the encoding modes of motion positional information, and the model training strategies. We evaluated the performance of all models on MSVD-QA [34], MSRVTT-QA [34], and MotionBench. Exp 0 is the final setting we utilized. Maximum values are in **bold**, and second-highest values are underlined.

Exp.	Fusion Type	Motion Position	Motion Warmup	MSVD-QA Acc./Score	MSRVTT-QA Acc./Score	MotionBench Lin.	Cur.	Rot.	Con.	Avg.
0	crossattn	pre-fusion	✓	74.3 / 4.0	57.0 / 3.5	59.8	42.7	47.7	46.6	**49.2**
1	w/o motion	-	✗	72.4 / 3.8	52.6 / 3.3	53.5	39.8	45.7	36.3	44.5
2	add	pre-fusion	✓	73.2 / 3.9	56.6 / 3.3	55.2	41.4	44.3	43.1	46.2
3	concat	pre-fusion	✓	73.4 / 3.9	56.7 / 3.3	56.1	42.3	45.6	44.1	47.0
4	w/o agg.	pre-fusion	✓	74.1 / 4.0	**57.2** / 3.5	58.8	42.5	48.6	46.2	49.0
5	crossattn	post-fusion	✓	**75.0** / 4.1	56.9 / 3.4	57.8	42.3	46.3	45.0	47.9
6	crossattn	no pos	✓	73.2 / 4.0	56.4 / 3.3	54.1	40.3	43.2	39.6	44.8
7	crossattn	pre-fusion	✗	73.2 / 4.0	56.3 / 3.2	54.5	40.3	44.6	40.8	45.6

Table 5. Ablation study on motion encoder layer number. 'w/o enc' means encoding without motion encoder. Maximum values are in **bold**, and second-highest values are underlined.

Encoder Layer	MSVD-QA Acc./Score	MSRVTT-QA Acc./Score	MotionBench Avg. Acc.
w/o enc	72.4 / 3.8	52.6 / 3.3	44.5
1	73.4 / 3.9	56.4 / 3.4	47.2
2	74.3 / 4.0	**57.2** / 3.5	**49.2**
3	**74.4** / 4.0	**57.2** / 3.5	48.8
4	74.1 / 4.0	57.0 / 3.5	49.0

achieves performance comparable to that of the full motion sequence while reducing computational costs. Consequently, we adopt a cross-attention fusion module with a pooling aggregator as our final design.

Motion Encoder Architecture. Since motion vectors are sparse relative to RGB frames, we employ a lightweight transformer to encode them. In Tab. 5, we present model performance across different numbers of layers in the motion encoder. The results show a slight performance improvement with more than two layers, suggesting that the sparse motion vectors do not require a complex module for effective encoding. Consequently, we select a two-layer transformer as the motion encoder.

Motion Position Encoding Motion vectors represent the movement from the previous frame to the current frame, creating a natural chronological order for the motion vector sequence within a GOP. To encode the temporal index of each motion vector, we use a learnable position embedding within the GOP. In Exp (0, 5-6) of Tab. 4, we compare various methods for incorporating temporal position embeddings into motion features. We examine three approaches: No Pos (Exp 6) encodes motion features without temporal position information, Pre-Fusion (Exp 0) adds learnable position embeddings to the motion encoder inputs, and Post-Fusion (Exp 5) adds learnable position embeddings to the motion encoder outputs. Our results show that the Pre-Fusion method achieves the best performance, indicating that early fusion supports more effective temporal modeling of motion sequences.

Motion Warmup Strategy Since the motion encoder is randomly initialized while the frame encoder is initialized with the pretrained vision backbone SigLIP-so400m [41], we employ a warm-up strategy to enhance the motion encoder. Specifically, we use motion vectors from the Something2Something-v2 dataset [7] along with their corresponding labels, applying a supervised training method with cross-entropy loss. This warm-up approach significantly improves model performance, as evidenced by the comparison between Exp 0 and 7 in Tab. 4.

5. Conclusion

In this work, we introduced **EMA**, a efficient motion-aware video understanding model for comprehensive video analysis by leveraging the native slow-fast structure of compressed video data. Our approach employs a motion-aware GOP encoder to effectively fuse spatial and motion information within compressed video units. This innovative encoding method integrates dense RGB spatial semantics with sparse motion dynamics, thereby reducing redundancy and inference costs while preserving robust video understanding capabilities. Additionally, we present MotionBench to address the gap in motion-focused evaluation of video models. Our model demonstrated superior performance across both public video benchmarks and MotionBench with less inference cost, underscoring its effectiveness in video comprehension. We believe that **EMA** and MotionBench represent important steps toward more efficient and accurate video understanding, with the potential to inspire future research that further investigates compressed video structures and motion-centric analysis within multimodal language models. We hope our research will inspire future advancements in the efficiency and motion awareness of video MLLMs.

6. Acknowledgement

This research is supported by Artificial Intelligence-National Science and Technology Major Project (2023ZD0121200) and the National Natural Science Foundation of China (6243000159, 62102416), and the Key Research and Development Program of Jiangsu Province under Grant BE2023016-3.

References

[1] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A frontier large vision-language model with versatile abilities. *arXiv preprint arXiv:2308.12966*, 2023. 7

[2] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 961–970, 2015. 7

[3] David Chen and William B Dolan. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies*, pages 190–200, 2011. 7

[4] Jiawei Chen and Chiu Man Ho. Mm-vit: Multi-modal video transformer for compressed video action recognition. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 1910–1921, 2022. 2, 3

[5] Yifan Du, Yuqi Huo, Kun Zhou, Zijia Zhao, Haoyu Lu, Han Huang, Wayne Xin Zhao, Bingning Wang, Weipeng Chen, and Ji-Rong Wen. Exploring the design space of visual context representation in video mllms. *arXiv preprint arXiv:2410.13694*, 2024. 2

[6] Chaoyou Fu, Yuhan Dai, Yondong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, et al. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv preprint arXiv:2405.21075*, 2024. 2, 6, 7

[7] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017. 5, 8

[8] Yubin Hu, Yuze He, Yanghao Li, Jisheng Li, Yuxing Han, Jiangtao Wen, and Yong-Jin Liu. Efficient semantic segmentation by altering resolutions for compressed videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22627–22637, 2023. 3

[9] Lianghua Huang, Yu Liu, Bin Wang, Pan Pan, Yinghui Xu, and Rong Jin. Self-supervised video representation learning by context and motion decoupling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13886–13895, 2021. 2, 3

[10] Yuqi Huo, Xiaoli Xu, Yao Lu, Yulei Niu, Mingyu Ding, Zhiwu Lu, Tao Xiang, and Ji-rong Wen. Lightweight action recognition in compressed videos. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 337–352. Springer, 2020. 3

[11] Yuqi Huo, Mingyu Ding, Haoyu Lu, Nanyi Fei, Zhiwu Lu, Ji-Rong Wen, and Ping Luo. Compressed video contrastive learning. *Advances in Neural Information Processing Systems*, 34:14176–14187, 2021. 2, 3

[12] Peng Jin, Ryuichi Takanobu, Wancai Zhang, Xiaochun Cao, and Li Yuan. Chat-univi: Unified visual representation empowers large language models with image and video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13700–13710, 2024. 1, 2, 6, 7

[13] Yang Jin, Zhicheng Sun, Kun Xu, Liwei Chen, Hao Jiang, Quzhe Huang, Chengru Song, Yuliang Liu, Di Zhang, Yang Song, et al. Video-lavit: Unified video-language pre-training with decoupled visual-motional tokenization. *arXiv preprint arXiv:2402.03161*, 2024. 2, 6, 7

[14] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 5

[15] Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What matters when building vision-language models? *arXiv preprint arXiv:2405.02246*, 2024. 5

[16] Bing Li, Jiaxin Chen, Xiuguo Bao, and Di Huang. Compressed video prompt tuning. *Advances in Neural Information Processing Systems*, 36:31895–31907, 2023. 3

[17] Jiapeng Li, Ping Wei, Yongchi Zhang, and Nanning Zheng. A slow-i-fast-p architecture for compressed video action recognition. In *Proceedings of the 28th ACM international conference on multimedia*, pages 2039–2047, 2020. 2, 3

[18] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding. *arXiv preprint arXiv:2305.06355*, 2023. 1, 2, 7

[19] KunChang Li, Yinan He, Yi Wang, Yizhuo Li, Wenhai Wang, Ping Luo, Yali Wang, Limin Wang, and Yu Qiao. Videochat: Chat-centric video understanding, 2023. 2

[20] Kunchang Li, Yali Wang, Yizhuo Li, Yi Wang, Yinan He, Limin Wang, and Yu Qiao. Unmasked teacher: Towards training-efficient video foundation models, 2023. 2

[21] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, et al. Mvbench: A comprehensive multi-modal video understanding benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22195–22206, 2024. 5, 7

[22] Yanwei Li, Chengyao Wang, and Jiaya Jia. Llama-vid: An image is worth 2 tokens in large language models. *arXiv preprint arXiv:2311.17043*, 2023. 1, 2, 6, 7

[23] Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *arXiv preprint arXiv:2311.10122*, 2023. 1, 6, 7

[24] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. 5

[25] Ruyang Liu, Chen Li, Yixiao Ge, Ying Shan, Thomas H Li, and Ge Li. One for all: Video conversation is feasible without video instruction tuning. *arXiv preprint arXiv:2309.15785*, 2023. 2, 7

[26] Ruyang Liu, Chen Li, Haoran Tang, Yixiao Ge, Ying Shan, and Ge Li. St-llm: Large language models are effective temporal learners. In *European Conference on Computer Vision*, pages 1–18. Springer, 2025. 7

[27] Ruipu Luo, Ziwang Zhao, Min Yang, Junwei Dong, Da Li, Pengcheng Lu, Tao Wang, Linmei Hu, Minghui Qiu, and Zhongyu Wei. Valley: Video assistant with large language model enhanced ability. *arXiv preprint arXiv:2306.07207*, 2023. 2, 5

[28] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*, 2023. 1, 2, 6, 7

[29] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1, 2

[30] Yaojie Shen, Xin Gu, Kai Xu, Heng Fan, Longyin Wen, and Libo Zhang. Accurate and fast compressed video captioning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15558–15567, 2023. 2, 3

[31] Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Xun Guo, Tian Ye, Yan Lu, Jenq-Neng Hwang, et al. Moviechat: From dense token to sparse memory for long video understanding. *arXiv preprint arXiv:2307.16449*, 2023. 1, 2, 7

[32] K Soomro. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012. 5

[33] Xiang Wang, Hangjie Yuan, Shiwei Zhang, Dayou Chen, Jiuniu Wang, Yingya Zhang, Yujun Shen, Deli Zhao, and Jingren Zhou. Videocomposer: Compositional video synthesis with motion controllability. *Advances in Neural Information Processing Systems*, 36, 2024. 3

[34] Dejing Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. Video question answering via gradually refined attention over appearance and motion. In *ACM Multimedia*, 2017. 1, 2, 6, 8

[35] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. Msr-vtt: A large video description dataset for bridging video and language. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5288–5296, 2016. 7

[36] Lin Xu, Yilin Zhao, Daquan Zhou, Zhijie Lin, See Kiong Ng, and Jiashi Feng. Pllava: Parameter-free llava extension from images to videos for video dense captioning. *arXiv preprint arXiv:2404.16994*, 2024. 2

[37] Mingze Xu, Mingfei Gao, Zhe Gan, Hong-You Chen, Zhengfeng Lai, Haiming Gang, Kai Kang, and Afshin Dehghan. Slowfast-llava: A strong training-free baseline for video large language models. *arXiv preprint arXiv:2407.15841*, 2024. 2

[38] Antoine Yang, Antoine Miech, Josef Sivic, Ivan Laptev, and Cordelia Schmid. Zero-shot video question answering via frozen bidirectional language models. *Advances in Neural Information Processing Systems*, 35:124–141, 2022. 1, 2, 7

[39] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024. 6

[40] Zhou Yu, Dejing Xu, Jun Yu, Ting Yu, Zhou Zhao, Yueting Zhuang, and Dacheng Tao. Activitynet-qa: A dataset for understanding complex web videos via question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9127–9134, 2019. 2, 6

[41] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023. 1, 6, 8

[42] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023. 1, 2, 7

[43] Peiyuan Zhang, Kaichen Zhang, Bo Li, Guangtao Zeng, Jingkang Yang, Yuanhan Zhang, Ziyue Wang, Haoran Tan, Chunyuan Li, and Ziwei Liu. Long context transfer from language to vision. *arXiv preprint arXiv:2406.16852*, 2024. 7

[44] Renrui Zhang, Jiaming Han, Chris Liu, Peng Gao, Aojun Zhou, Xiangfei Hu, Shilin Yan, Pan Lu, Hongsheng Li, and Yu Qiao. Llama-adapter: Efficient fine-tuning of language models with zero-init attention. *arXiv preprint arXiv:2303.16199*, 2023. 1, 7

[45] Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Video instruction tuning with synthetic data. *arXiv preprint arXiv:2410.02713*, 2024. 2