Chat with AI: The Surprising Turn of Real-time Video Communication from Human to AI

Jiangkai Wu, Zhiyuan Ren, Liming Liu, Xinggong Zhang
Peking University

Abstract

AI Video Chat emerges as a new paradigm for Real-time Communication (RTC), where one peer is not a human, but a Multimodal Large Language Model (MLLM). This makes interaction between humans and AI more intuitive, as if chatting face-to-face with a real person. However, this poses significant challenges to latency, because the MLLM inference takes up most of the response time, leaving very little time for video streaming. *Due to network uncertainty, transmission latency becomes a critical bottleneck preventing AI from being like a real person.* To address this, we call for AI-oriented RTC research, *exploring the network requirement shift from "humans watching video" to "AI understanding video".* We begin by recognizing the main differences between AI Video Chat and traditional RTC. Then, through prototype measurements, we identify that ultra-low bitrate is a key factor for low latency. To reduce bitrate dramatically while maintaining MLLM accuracy, we propose *Context-Aware Video Streaming* that recognizes the importance of each video region for chat and allocates bitrate almost exclusively to chat-important regions. To evaluate the impact of video streaming quality on MLLM accuracy, we build the first benchmark, named ***De**graded **Vi**deo Understanding **Bench**mark (**DeViBench**)*. Finally, we discuss some open questions and ongoing solutions for AI Video Chat. DeViBench is open-sourced at: https://github.com/pku-netvideo/DeViBench.

CCS Concepts

• **Computing methodologies** → **Artificial intelligence**; • **Networks** → **Application layer protocols**; • **Information systems** → **Multimedia streaming**; • **Human-centered computing** → **Human computer interaction (HCI)**.

HotNets '25, College Park, MD, USA

ACM ISBN 979-8-4007-2280-6/2025/11
https://doi.org/10.1145/3772356.3772390

Keywords

Real-time Communication, Generative AI, Multimodal Large Language Model, AI Video Chat, Latency, Benchmark

ACM Reference Format:
Jiangkai Wu, Zhiyuan Ren, Liming Liu, Xinggong Zhang. 2025. Chat with AI: The Surprising Turn of Real-time Video Communication from Human to AI. In *The 24th ACM Workshop on Hot Topics in Networks (HotNets '25), November 17–18, 2025, College Park, MD, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3772356.3772390

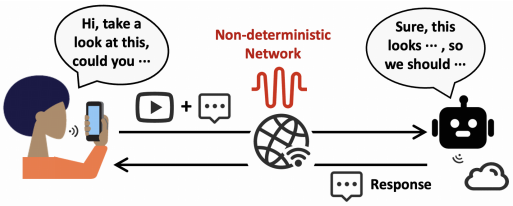

Figure 1: AI Video Chat is a new paradigm for real-time communication. The user sends video and audio to the AI for thinking. The AI feeds back to the user. Low latency is crucial for making AI act like a real person.

1 Introduction

AI Video Chat is a new paradigm for Real-time Communication (RTC). Since OpenAI released GPT-4o [15], Multimodal Large Language Models (MLLMs) have been continuously emerging, such as Qwen2.5-Omni [40], Gemini-1.5 [32], VITA-1.5 [13], and OmniLive [46]. Compared to traditional LLMs, MLLMs enable users to directly input video and audio for interaction, rather than just text. This makes the interaction between humans and AI more intuitive as if chatting face-to-face with a real person [7]. However, MLLMs require high-performance computing devices to support real-time inference (such as 8*A100 GPUs [27]). Mobile devices (like phones or smart glasses) cannot meet the computing requirements, which makes MLLMs inevitably deployed in the cloud. So in existing systems, the client sends user video and audio to the cloud for MLLM inference, and the cloud then feeds back responses to users, as shown in Figure 1.

AI Video Chat raises significant challenges to RTC transmission latency. To ensure a fluent interactive experience, the response latency of video chat needs to remain below 300 ms [18]. In traditional video chat, the human peer on the other side can respond instantly. Therefore, response latency

largely stems from RTC's end-to-end latency (including capture, transmission, decoding, and playback buffer latencies). Among them, transmission latency is sensitive to network conditions and continuously increases when the network deteriorates. To reduce transmission latency, current state-of-the-art RTC frameworks (such as WebRTC [6]) adopt technologies like Adaptive Bitrate (ABR) [3, 14, 16, 23, 41], Congestion Control [5, 11, 12, 25], and Forward Error Correction (FEC) [4, 10, 20, 24, 29] to satisfy user experience. However, in AI Video Chat, responses are generated through MLLM in an autoregressive manner, which is time-consuming. Even when inputting only audio tokens, the computational latency is at least 232 ms [15]. To constrain the response latency below 300 ms, even ignoring other latencies in the RTC pipeline, the time left for transmission is at most 68 ms, which is difficult to guarantee. So the large latency makes users clearly feel that the other side is not a real person.

Is it possible to reduce the latency of AI Video Chat to an extremely low level? This vision allows us to grasp the "holy grail" [7] of AI research from the perspective of network systems: making AI like real humans.

To realize this vision, we call for AI-oriented RTC research, exploring the network requirement shift from "humans watching video" to "AI understanding video". We begin by recognizing the main differences between AI Video Chat and traditional RTC: *First, QoE changes from human perceptual quality to MLLM response accuracy. Second, jitter has no impact. Third, the receiver throughput is far lower than the sender throughput. Fourth, uplink is more pressing than Downlink.* Then, through prototype measurements, we identify a key factor for low latency: *ultra-low bitrate*. Based on two insights, we make contributions:

- **Video should be Context-Aware (§3.2).** To reduce bitrate dramatically while maintaining MLLM accuracy, we propose *Context-Aware Video Streaming*, allocating more bitrate to chat-important video regions while allocating as little bitrate as possible to chat-irrelevant regions.
- **The First Benchmark (§3.1).** Considering that there is no benchmark that can evaluate the impact of video quality on MLLM accuracy, we propose the first one, named ***De**graded **Vi**deo Understanding **Bench**mark* (***DeViBench***).

2 Motivation

2.1 Main differences between AI Video Chat and traditional RTC

QoE changes from human perceptual quality to MLLM response accuracy. In traditional RTC, QoE focuses on measuring the perceptual quality of human eyes. For example, using penalty terms like stalling time [10, 41] or quality variance [8] to measure temporal stability. Using SSIM [41] or VMAF [2] to measure visual quality. But in AI Video Chat, the viewer of the video changes from humans to MLLMs. At this point, most perception-related metrics are no longer needed, and the optimization objective of QoE becomes the accuracy and latency of MLLM's responses. So the RTC strategy can undergo a significant turn. For example, to reduce latency, temporal stability (such as frequent bitrate adjustments) and visual quality (like lowering bitrate in Figure 4) can be sacrificed, as long as accuracy is enough.

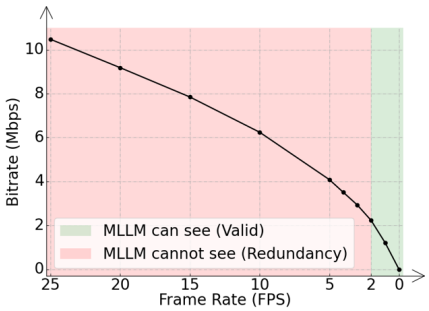

Figure 2: MLLM processes video at a very low frame rate (green), so most frames are redundancy (red).

Jitter has no impact. Due to network uncertainties (congestion, packet loss, etc.), even when the sender transmits frames at fixed time intervals, the time intervals of received frames will experience jitter. Direct playback will result in uneven video speed, causing stuttering. Therefore, traditional RTC employs a jitter buffer [47], trading latency for smoothness. For MLLMs, the difference is that their perception of time does not rely on real physical time, but rather on positional encoding computation [13, 40, 46]. Positional encoding is only associated with the frame's capture timestamp and unrelated to the actual receiving time; thus, jitter has no impact on the MLLM's perception of the video. It means that in AI video chat, the buffer can be removed to reduce the latency.

The receiver throughput is far lower than the sender throughput. In traditional RTC, the data throughput at the receiver is comparable to that at the sender, for example, both are 1920*1080 resolution at a 30 FPS frame rate. However, in AI Video Chat, MLLM is limited by context length (finite number of tokens) and real-time inference, and cannot fully process the received video. Therefore, the received video needs to be actively downsampled before being fed to the MLLM. In terms of frame rate, existing AI Video Chat systems support a maximum processing rate of only 2 FPS [13, 38, 40]. In terms of resolution, regardless of how high the original resolution is, it will be downsampled to no more than 602,112 pixels [40]. So traditional RTC contains massive redundancy that MLLMs cannot perceive, as shown in Figure 2.

Uplink is more pressing than Downlink. In traditional RTC, each peer is both video sender and receiver. In contrast, AI Video Chat is unidirectional video transmission, where the user only acts as the video sender and MLLM only acts as

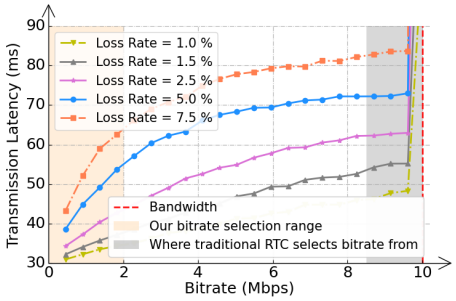

Figure 3: How bitrate and packet loss affect latency (with 10 Mbps bandwidth). To optimize video quality, traditional RTC systems select bitrate from the gray region. But in AI video chat, to maintain accuracy, we only need to select bitrate from the yellow region (§2.2).

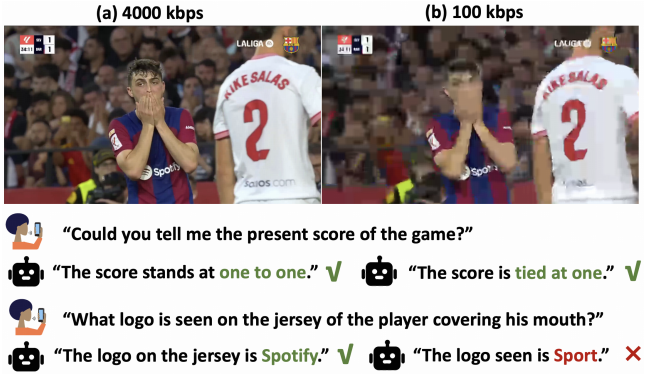

Figure 4: Why video should be context-aware in AI Video Chat. In the first dialogue, even if the video bitrate decreases from 4000 Kbps to 200 Kbps, the MLLM can still response accurately. But in the second dialogue from StreamingBench [22], the blurry video leads to incorrect responses. Thus, rather than reducing bitrate in a context-agnostic manner, bitrate allocation should be determined by the current chat context (§2.3).

the video receiver. MLLM sends responses to the user in the form of audio or text, and these representations have much lower bitrates than video. Thus, uplink needs better network conditions than downlink, for example, larger bandwidth.

2.2 What factors affect the transmission latency of AI Video Chat?

To analyze the factors affecting transmission latency in AI Video Chat, we built a prototype and conducted preliminary measurements. Specifically, we implement a WebRTC-based unidirectional video transmission system and a network emulator. Under given bandwidth (10 Mbps) and one-way network delay (30 ms), we run video transmission for a total duration of 40,489 seconds, and collect statistics on transmission latency (the time from the frame being sent to being completely received, excluding the jitter buffer §2.1) with different packet loss rates and bitrates, as shown in Figure 3:

First, when the bitrate exceeds the bandwidth, transmission latency becomes enormous. This is because excessive bitrate causes congestion, where packet accumulation causes latency to increase rapidly. Therefore, existing RTC systems employ ABR algorithms to set the bitrate as close as possible to (but below) the bandwidth, maximizing video quality while avoiding stalling, as shown in the **grey region** of Figure 3. Second, when the bitrate does not exceed the bandwidth, transmission latency also increases as the bitrate increases. This is due to the fact that each packet has a limited size (around 1400 bytes). A higher bitrate means each frame will be divided into more packets. Due to packet loss, more packets mean the probability of each frame being completely received in one attempt decreases. For packets that are not received, retransmission may be required, potentially leading to increased latency. Therefore, even when the bitrate is below the bandwidth, AI Video Chat can further reduce the bitrate to achieve lower latency. **This differs from traditional ABR and offers another space for bitrate selection**, as shown in the **yellow region** of Figure 3.

2.3 Key Insights and Potential Gains

Video should be Context-Aware. According to §2.2 reducing video bitrate can decrease transmission latency. To reduce bitrate, existing methods typically increase quantization parameters [30], which inevitably degrades video quality. *Interestingly, the degradation in video quality does not necessarily lead to a decrease in MLLM accuracy, which depends on the current chat context.* As illustrated in Figure 4, when the user asks "Could you tell me the present score of the game?", even if the video bitrate is reduced from 4000 Kbps to 200 Kbps, the MLLM can still answer accurately. However, when the user asks "What logo is seen on the jersey of the player covering his mouth?", the blurry video leads to incorrect responses. This is because, in different chat contexts, the MLLM needs to focus on different video regions. Meanwhile, different video regions are affected differently by low bitrate. Thus, rather than reducing bitrate in a context-agnostic manner, the video should be context-aware. More bitrate should be allocated to chat-important regions, while less bitrate should be allocated to chat-irrelevant regions.

How to be aware of the chat context? Our idea is: the user words can indicate which video regions are important for the current chat. Therefore, we can take the user words as a reference to compute the semantic correlation of different video regions. For this, we adopt the Contrastive Language-Image Pre-Training (CLIP) model [28], which maps images and language to the same feature space. Hence, to derive

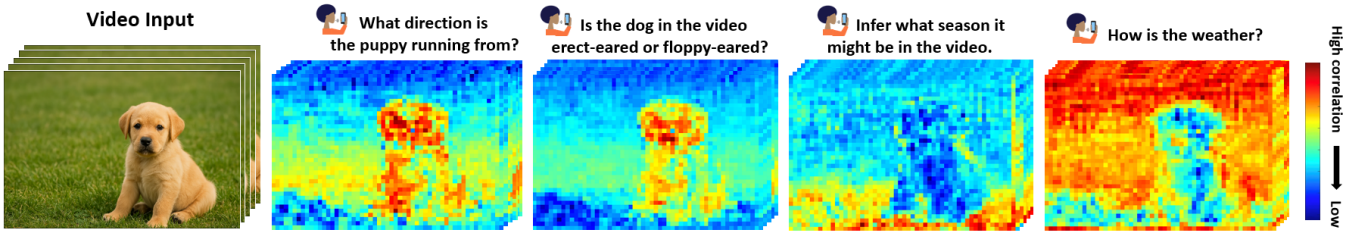

Figure 5: How to achieve context awareness? The user words can indicate which regions in the video are important for the current chat context. Based on CLIP, we can even recognize important regions through high-level understanding. For example, in the third dialogue, the growth of grass implies the current season (§2.3).

semantic correlation, we only need to compute the similarity of features between video regions and user words. We show some examples in Figure 5, which demonstrates that user words and CLIP can well point out the importance of different video regions for chatting. For example, when the user asks "Is the dog in the video erect-eared or floppy-eared?", the dog's head region exhibits the highest correlation. On the other hand, even when the user words do not explicitly indicate the object, CLIP can still estimate correlation based on high-level understanding. For example, when the user asks "Infer what season it might be in the video", grass has the highest correlation. This is because the growth of grass can imply the current season (CLIP even ignores the blurry grass in the distance). Thus, this context-aware mechanism allows us to optimally allocate the bitrate (**§3.2**).

The first benchmark evaluates how video streaming quality affects MLLM accuracy. According to §2.1, QoE metrics in AI Video Chat change from perception to accuracy. This causes existing benchmarks in the video streaming field to be inapplicable, as they focus on perceptual quality and do not involve response accuracy. In the MLLM field, there are some benchmarks targeting Streaming Video Understanding tasks [38, 42], such as StreamingBench [22]. In these benchmarks, each video includes several Question-Answer (QA) samples for evaluating the response accuracy. However, these benchmarks aim to test the MLLM's intelligence, so all the input videos are ideally high-bitrate (e.g., 4000 Kbps).

To evaluate how video streaming quality affects accuracy, we transcode videos from StreamingBench to 200 Kbps. Then we conduct testing on these low-bitrate videos with the original QA samples. The results show that only **8%** of QA samples are answered incorrectly at low bitrate and correctly at high bitrate. This is because the QA samples in StreamingBench are too simple and high-level, requiring only coarse-grained video content to answer correctly. For example, in Figure 4, when the question is "What is the player doing?", even if the video quality is particularly poor, the MLLM can still provide the correct answer "He is covering his mouth." However, in real-world scenarios, there are often many detail-rich questions that are very sensitive to video quality. For example, in Figure 4, when the question is "How many spectators can be seen?", even slight blurriness will prevent the MLLM from providing the correct answer. So it is necessary to establish a more challenging benchmark to reflect the real-world impact of video degradation on MLLM accuracy (**§3.1**).

3 Towards RTC for AI: Case Study

We begin by constructing the first benchmark evaluating how video quality affects MLLM accuracy, named **De**graded **Vi**deo Understanding **Bench**mark (**DeViBench**). Then we present a case study: **Context-Aware Video Streaming**.

3.1 DeViBench

In this section, we propose DeViBench. As described in §2.3, we need to construct QA samples that are sensitive to video quality. For this, the most straightforward way is to hire volunteers to ask tricky questions about degraded videos. However, this is too expensive and inefficient, hindering the scale-up of the dataset. **So we ask: Can QA samples be constructed automatically and cheaply?** Rethinking the background of AI Video Chat, MLLMs are already capable of understanding videos and giving responses. So we leverage MLLMs to replace human volunteers and develop an automatic QA sample construction pipeline. As illustrated in Figure 6, this pipeline consists of 5 steps:

Video Collection. We first collect videos to ask questions. To align with the domain and scale of existing MLLM benchmarks [22], we directly use their videos (discarding QA).

Video Preprocessing. To allow MLLMs to understand the quality degradation caused by low bitrates, we transcode the original videos to low-bitrate versions (200 Kbps) using x265 from ffmpeg version N-118035-gc1e3d55f99. The low bitrate video and the original video are horizontally concatenated into one video. Then this concatenated video will be input to the MLLM for understanding and QA generation.

QA Generation. To enable MLLMs to generate QA samples based on the concatenated video, we carefully designed a prompt with guidance from persona, context, core task, execution steps, constraints, and output format, as shown in

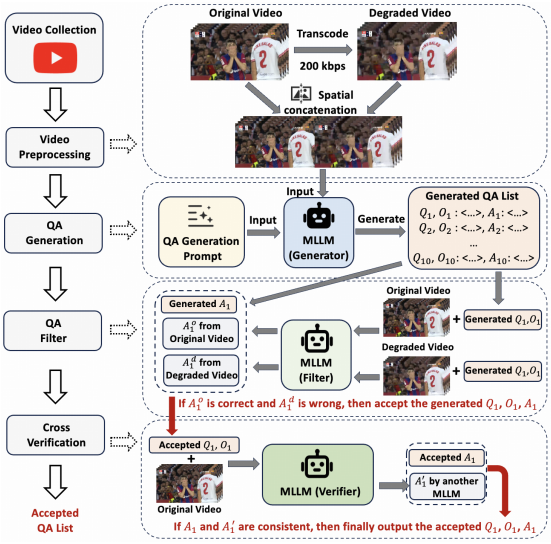

Figure 6: DeViBench's pipeline for automatic QA sample construction. Details can be found in §3.1.

Prompt used for **QA Sample Generation**

Persona: You are an expert Visual Question Answering (VQA) system designer, specializing in perceiving fine-grained details.
Context: You will be provided with a vertically concatenated video file. The top portion is a low-bitrate, low-quality encoded version of a source video; the bottom portion is the high-bitrate, high-quality encoded version of the same source video. Both parts are perfectly synchronized in content.
Core Task: Your objective is to generate a "quality-sensitive" multiple-choice question. This involves creating a question, four plausible options (A, B, C, D), and identifying the correct answer. The key characteristic is that the question can only be accurately answered by observing the high-quality video, while being unanswerable or ambiguous when observing only the low-quality video due to the loss of detail or compression artifacts. *Specially, In this task, you should focus on the dynamic details, that is, the answer to the question cannot be derived solely from one frame of the video.*
Execution Steps:
Identify Discrepancy: First, in your internal thought process, pinpoint a visual element that is clearly visible in the high-bitrate video (bottom) but becomes blurred, distorted, or completely lost in the low-bitrate video (top). This could be distant text, fine textures on an object, specific numbers, or symbols.
Formulate Question: Design a question that revolves around this identified "point of difference."
Generate Options and Answer:

- ***Correct Answer:*** Determine the single correct answer based on the high-bitrate video (bottom). This will be one of the four options.
- ***Distractors:*** Create three plausible but incorrect options. Crucially, at least one of these distractors should be a potential misinterpretation from the low-quality video (e.g., what the blurry text might look like, or a shape that an artifact resembles). All four options should be of the same type and format (e.g., all numbers, all words, all phrases).

Constraints:
Wording: The question and options must be phrased neutrally and universally, as if you have only seen a single video source. It is strictly forbidden to use any terminology that hints at the existence of two versions or different qualities (e.g., avoid "the bottom half," "the high-quality version," "the clearer part," etc.).
Focus: The question should focus on an objective, concrete fact, not a subjective interpretation.

Output Format: Please provide the final result strictly in the following format:
Question: [Your formulated question here]
Options: A) [Option A] B) [Option B] C) [Option C] D) [Option D]
Answer: [Correct Option Letter, e.g., C]

Figure 7: Our prompt for QA Sample Generation.

Figure 7. This prompt ensures that MLLMs can recognize quality differences and generate quality-sensitive QA samples. To facilitate judging whether the answer is correct, we generate multiple-choice questions with four options (A, B, C, D). Users can directly calculate accuracy by matching the answer letters, without needing to measure semantic consistency. We also encourage MLLMs to generate questions that require at least multiple frames to answer, in order to enhance the temporal dependency of the questions. Qwen3-VL-plus thinking [1] is adopted as the generator.

QA Filtering. The generated QA pairs will be filtered. We separately input the original video and the low bitrate video into the MLLM and use the generated QA pairs for questioning. If the answer from the original video is correct and the answer from the low bitrate video is wrong, we accept this QA pair. In practice, Qwen2.5-Omni [40] is adopted as the filter and **11.16%** of the QA pairs can be accepted.

Cross Verification. Since the answer generated by MLLM may also be incorrect, this cannot be filtered out through the above testing. Hence, we utilize another MLLM for cross-verification. We feed the above accepted question into another MLLM, and if the new answer is consistent with the above accepted answer, we finally approve this QA pair. In our experiments, GLM-4.5V thinking [33] is adopted as the verifier and **70.61%** of the accepted QA pairs can pass cross-verification. Considering all the above validations together, finally **7.8%** of the generated QA pairs are valid.

Finally, we produce 1,074 QA samples, with details summarized in Table 1, including QA types, total duration, total money spent, and total time cost. We also analyze the distribution of different QA types, as shown in Figure 8. In terms of categories, there are text-rich understanding (54.84%), action perception (17.03%), attribute perception (14.43%), counting (6%), object perception (5.9%), and spatial understanding (1.8%). In terms of temporal dependency, 34.45% of the questions necessitate multiple frames for answering, whereas 65.55% are answerable with a single frame. To confirm whether these MLLM-generated QA samples are usable, we spot-check 100 QA samples for manual answering. Among them, 95% of the generated questions are answerable by humans, and 84% of the generated answers are correct.

Table 1: Benchmark summary

Number of QA samples	1,074
QA sample types	6*2
Total duration (s)	180,000
Total money spent ($)	68.47
Total time cost (s)	99,471

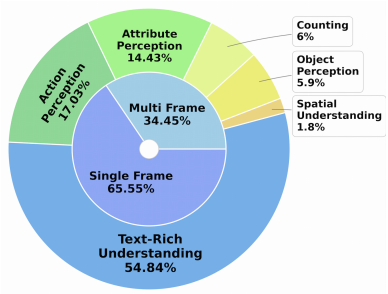

Figure 8: Distribution of our generated QA samples. Outer ring: QA categories. Inner ring: Whether the question requires multiple frames to answer.

3.2 Context-Aware Video Streaming

In this section, we describe how to achieve context-aware streaming, significantly reducing bitrate while maintaining MLLM accuracy. According to §2.3, we first leverage the

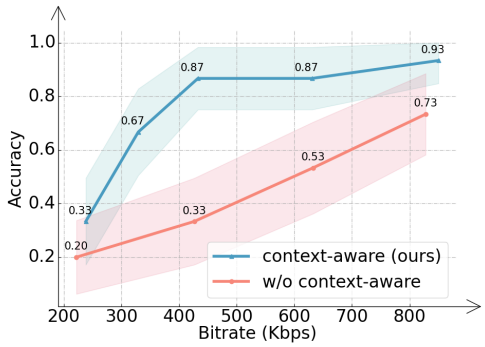

Figure 9: Context-aware streaming can dramatically lower the bitrate while maintaining MLLM accuracy.

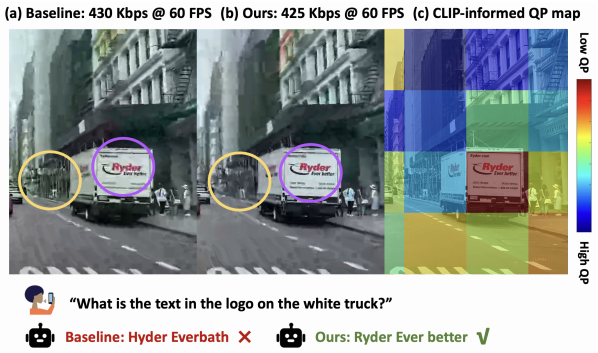

Figure 10: An example of accuracy gain. We visualize the frame input to the MLLM. (a) Encoded with default settings. (b) Encoded with CLIP-informed QP. (c) The CLIP-informed QP map. The results show that even with similar bitrates (430 Kbps vs. 425 Kbps), ours allocates more bits to chat-important regions (e.g., purple circles) and fewer bits to chat-irrelevant regions (e.g., yellow circles), thus improving MLLM accuracy.

CLIP model to compute the semantic correlation between user words and video regions. To ensure real-time computing on mobile devices, we adopt Mobile-Clip [35]. Specifically, given the current user words $\mathcal{T}$ and the latest video frame $F \in \mathbb{R}^{H\times W\times 3}$, we first partition F into non-overlapping patches $\{P_{mn} \mid 1 \le m \le \lfloor H/N \rfloor, 1 \le n \le \lfloor W/N \rfloor\}$, where each patch $P_{mn} \in \mathbb{R}^{N\times N\times 3}$ represents a video region. Then, the CLIP visual encoder $\phi_v(\cdot) : \mathbb{R}^{N\times N\times 3} \to \mathbb{R}^d$ is employed to extract patch-wise features $f^v_{mn} = \phi_v(P_{mn})$, while the CLIP language encoder $\phi_l(\cdot) : \mathcal{T} \to \mathbb{R}^d$ encodes the user words $\mathcal{T}$ into a semantic features $f^l = \phi_l(\mathcal{T})$. Here d denotes the unified feature dimension. Semantic correlation ρ_{mn} between user words and patches is then computed as cosine similarities:

$$\rho_{mn} = \frac{f^v_{mn} \cdot f^l}{\|f^v_{mn}\| \left\|f^l\right\|} \in [-1, 1] \tag{1}$$

Semantic correlation ρ_{mn} can measure the importance of region P_{mn} for the current chat context. The larger ρ_{mn} is, the more important P_{mn} is. So we can allocate more bitrate to important regions while allocating as little bitrate as possible to irrelevant regions. To achieve this, we adjust the Quantization Parameters (QP) of different regions during video encoding. When QP is larger ($0 \le \text{QP} \le 51$), the region occupies less bitrate, but the quality becomes worse. Specifically, for region P_{mn}, its QP_{mn} is derived as:

$$\text{QP}_{mn} = 51 \left(1 - \left(\frac{\rho_{mn} + 1}{2}\right)^{\gamma}\right) \tag{2}$$

Where γ is the temperature coefficient, set to 3 to aggressively penalize irrelevant regions ($\rho_{mn} \ll 1$). To achieve fine-grained QP control, we adopt H.265 implemented by Kvazaar [36] to encode ours and baseline. Except for the QP values, ours and baseline use the same encoding parameters. The frame rate is consistent with the video source (e.g., 60 FPS). The specific Kvazaar command lines can be found in our open-source link. As for decoding, both ours and baseline adopt x265 from decord 0.6.0, maintaining the same decoding parameters. We test with Qwen2.5-Omni [40]. Code and model are frozen before testing. We keep the default settings (the same random seed, system prompt, and configuration as officially recommended), without tuning for the QA samples.

We evaluate the performance gains [1] in Figure 9. The results demonstrate that context-aware streaming can dramatically lower the bitrate while maintaining MLLM accuracy. For example, when the bitrate is reduced from **827.9** Kbps to **426.4** Kbps (**48.5%** reduction), the MLLM accuracy drops from **0.73** to **0.33**. After integrating context-aware streaming, as the bitrate drops from **850.1** Kbps to **432.7** Kbps, the accuracy only decreases from **0.93** to **0.87**. To intuitively demonstrate the benefits, we visualize two sampled frames fed into the MLLM, as shown in Figure 10. The results show that even with similar bitrates (430 Kbps vs. 425 Kbps), our method allocates more bits to chat-important regions (e.g.,

[1] Since DeViBench is continuously scaling up and iterating, the experiments in Figure 9 are frozen at an earlier version (small-scale, free-response, also available in the open-source link), rather than the version reported in §3.1. This is because video encoding takes a long time to run, and time constraints prevented us from rerunning the experiments on the current version. During Kvazaar encoding, the target bitrate often differs greatly from the actual bitrate. So we use a trial-and-error approach to ensure that the actual bitrates of ours and the baseline are comparable. Each video requires many encoding iterations, causing large time costs. Despite not updating the experiments, we speculate that the baseline's accuracy would be higher on the current version. This is because, compared to the previous free-response questions, multiple-choice questions are easier to answer. On one hand, the options, as part of the question prompt, provide sufficient hints to the MLLMs. On the other hand, even if the video is too blurry to see clearly, MLLMs can still make a vague guess from the ABCD options (with at least 25% accuracy).

purple circles) and fewer bits to chat-irrelevant regions (e.g., yellow circles), thus improving MLLM accuracy.

4 DISCUSSIONS AND OPEN QUESTIONS

Proactive context-aware. In this paper, we leverage user words to achieve context awareness. It may not necessarily perform well in practice. Because it requires user words to be known before video encoding. But users may speak at any moment in the video, causing user words not to cover some segments. For example, in some benchmarks like [22, 38, 42], they assume that users ask questions at the end of the video. As our next step, we are building a *proactive context-aware mechanism* that can actively recognize important video regions even if users do not speak.

MLLM long-term memory. To minimize bitrate, the sender discards most video content irrelevant to the current chat context. This is based on the assumption that the current chat only references real-time video content. However, some MLLMs have developed long-term memory mechanisms [26, 37, 39], allowing chats to reference historical video content. Some video content, even if not relevant in the current chat context, may be needed in future chats. As our next step, we are developing a *semantic layered video streaming framework*. Different from SVC [30] that layers based on video quality, we layer by semantic correlation. The base layer contains the most important video content for the current chat context, so it must ensure low latency. The enhancement layers contain complete video details, used to offline build long-term memory, so they are not sensitive to latency.

Token pruning. To further lower the end-to-end latency, it is necessary to reduce the inference latency of MLLMs. Since MLLMs run in an autoregressive manner, a straightforward solution is to decrease the number of input tokens. Some related work exploits attention mechanisms [48] or video redundancy [43] to prune most visual tokens, without affecting MLLM accuracy. In this paper, context-aware streaming has already recognized important video regions, so it makes much sense to prune tokens from chat-irrelevant regions. As our next step, we are developing *context-aware token pruning mechanisms* to accelerate MLLM inference.

Client-side computation. Despite being optimized for mobile devices, Mobile-CLIP still incurs considerable computation. This leads to the computational resources not being fairly equalized in the comparison, because the encoders for both ours and the baseline use the default preset (medium). As future work, the baseline can adopt a more complex encoder preset (e.g., slower) to balance computational resources and achieve better video quality. Moreover, extra computational resources can also be used to explore model collaboration. For example, deploying a mobile MLLM [17, 44] on the client to handle simple questions locally, while only transmitting challenging videos to the cloud-side MLLM.

Client-side tokenizer and token streaming. Is it possible to offload the video tokenizer from the server to the client and stream video tokens to the MLLM? This offers three potential gains: First, the tokenizer can serve as a powerful video compressor. For instance, MAGVIT-v2 [45] achieves a better compression ratio than H.266 [19]. Second, video tokens are loss-resilient. On one hand, even when 82.8% of tokens are lost, the MLLM can still maintain 98% of its original accuracy. On the other hand, missing tokens can be recovered at the receiver using some Masked Language Models [20]. Third, offloading the tokenizer can fully leverage client-side computational resources, thereby alleviating server-side pressure and increasing the number of concurrent requests. However, *despite the significant potential benefits, this approach is infeasible.* It is important to note that there are two types of video tokens: continuous tokens (outputs from the encoder, represented as embeddings) and discrete tokens (quantized through a codebook, represented as indices [34]). Only discrete tokens have a low bitrate [45], while continuous tokens are uncompressed floating-point tensors whose bitrate is too high to stream. Discrete tokens are used only for AIGC tasks (such as text-to-video generation [9, 21, 45]), while MLLMs exclusively employ continuous tokens for video understanding [9, 21, 40]). Although some earlier MLLMs adopted discrete tokens [31], state-of-the-art MLLMs no longer do so due to the significant accuracy loss caused by quantization.

Circularity. The substantial gains shown in Figure 9 partially stem from circularity. Since DeViBench has "cherry-picked" QA samples on which the MLLM makes mistakes at 200 Kbps, the baseline's low accuracy is expected (Although different encoders were used during testing and QA selection). In fact, this is the motivation for proposing DeViBench. As discussed in §2.3, only 8% of existing QA samples exhibit errors at low bitrates and 92% can be answered correctly. This indicates that existing QA samples are too coarse-grained and lack reference to details. To bridge this gap, we need more video quality-sensitive QA samples to evaluate how quality degradation affects MLLM accuracy. In future work, we will analyze the proportion of such video quality-sensitive QA in real-world AI Video Chat applications. On the other hand, we will also test the accuracy of various MLLMs on such QA samples to demonstrate the generality of the gains.

Acknowledgements

We sincerely thank our shepherd Keith Winstein, and reviewers for their valuable feedback. This work is sponsored by the National Natural Science Foundation of China (62431017). We gratefully acknowledge the support of Key Laboratory of Intelligent Press Media Technology. Xinggong Zhang is the corresponding author (zhangxg@pku.edu.cn).

References

[1] 2025. Qwen3-VL-Plus. https://bailian.console.aliyun.com/?spm=a2c4g.11186623.0.0.74e555efL5VoGI&tab=model#/model-market/detail/qwen3-vl-plus.

[2] 2025. VMAF. https://github.com/Netflix/vmaf.

[3] Zahaib Akhtar, Yun Seong Nam, Ramesh Govindan, Sanjay Rao, Jessica Chen, Ethan Katz-Bassett, Bruno Ribeiro, Jibin Zhan, and Hui Zhang. 2018. Oboe: Auto-tuning video ABR algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*. 44–58.

[4] Congkai An, Huanhuan Zhang, Shibo Wang, Jingyang Kang, Anfu Zhou, Liang Liu, Huadong Ma, Zili Meng, Delei Ma, Yusheng Dong, et al. 2025. Tooth: Toward Optimal Balance of Video QoE and Redundancy Cost by Fine-Grained FEC in Cloud Gaming Streaming. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*. 635–651.

[5] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. 2017. BBR: Congestion-based congestion control. *Commun. ACM* 60, 2 (2017), 58–66.

[6] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and design of the Google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International Conference on Multimedia Systems*. 1–12.

[7] Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. 2024. VideoLLM-Online: Online video large language model for streaming video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18407–18418.

[8] Tianyu Chen, Yiheng Lin, Nicolas Christianson, Zahaib Akhtar, Sharath Dharmaji, Mohammad Hajiesmaili, Adam Wierman, and Ramesh K Sitaraman. 2024. SODA: An adaptive bitrate controller for consistent high-quality video streaming. In *Proceedings of the ACM SIGCOMM 2024 Conference*. 613–644.

[9] Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. 2025. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811* (2025).

[10] Yihua Cheng, Ziyi Zhang, Hanchen Li, Anton Arapin, Yue Zhang, Qizheng Zhang, Yuhan Liu, Kuntai Du, Xu Zhang, Francis Y Yan, et al. 2024. GRACE: Loss-Resilient Real-Time video through neural codecs. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 509–531.

[11] Mo Dong, Qingxi Li, Doron Zarchy, P Brighten Godfrey, and Michael Schapira. 2015. PCC: Re-architecting congestion control for consistent high performance. In *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. 395–408.

[12] Mo Dong, Tong Meng, Doron Zarchy, Engin Arslan, Yossi Gilad, Brighten Godfrey, and Michael Schapira. 2018. PCC vivace: Online-Learning congestion control. In *15th USENIX symposium on networked systems design and implementation (NSDI 18)*. 343–356.

[13] Chaoyou Fu, Haojia Lin, Xiong Wang, Yi-Fan Zhang, Yunhang Shen, Xiaoyu Liu, Haoyu Cao, Zuwei Long, Heting Gao, Ke Li, et al. 2025. Vita-1.5: Towards GPT-4o level real-time vision and speech interaction. *arXiv preprint arXiv:2501.01957* (2025).

[14] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM*. 187–198.

[15] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. GPT-4o system card. *arXiv preprint arXiv:2410.21276* (2024).

[16] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with Festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. 97–108.

[17] Zhiwei Jin, Xiaohui Song, Nan Wang, Yafei Liu, Chao Li, Xin Li, Ruichen Wang, Zhihao Li, Qi Qi, Long Cheng, Dongze Hao, Quanlong Zheng, Yanhao Zhang, Haobo Ji, Jian Ma, Zhitong Zheng, Zhenyi Lin, Haolin Deng, Xin Zou, Xiaojie Yin, Ruilin Wang, Liankai Cai, Haijing Liu, Yuqing Qiu, Ke Chen, Zixian Li, Chi Xie, Huafei Li, Chenxing Li, Chuangchuang Wang, Kai Tang, Zhiguang Zhu, Kai Tang, Wenmei Gao, Rui Wang, Jun Wu, Chao Liu, Qin Xie, Chen Chen, and Haonan Lu. 2025. AndesVL Technical Report: An Efficient Mobile-side Multimodal Large Language Model. arXiv:2510.11496 [cs.CV] https://arxiv.org/abs/2510.11496

[18] Zeqi Lai, Weisen Liu, Qian Wu, Hewu Li, Jingxi Xu, and Jianping Wu. 2022. SpaceRTC: Unleashing the low-latency potential of mega-constellations for real-time communications. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 1339–1348.

[19] Minhun Lee, HyeonJu Song, Jeeyoon Park, Byeungwoo Jeon, Jungwon Kang, Jae-Gon Kim, Yung-Lyul Lee, Je-Won Kang, and Donggyu Sim. 2023. Overview of versatile video coding (H. 266/VVC) and its coding performance analysis. *IEIE Transactions on Smart Processing & Computing* 12, 2 (2023), 122–154.

[20] Tianhong Li, Vibhaalakshmi Sivaraman, Pantea Karimi, Lijie Fan, Mohammad Alizadeh, and Dina Katabi. 2023. Reparo: Loss-resilient generative codec for video conferencing. *arXiv preprint arXiv:2305.14135* (2023).

[21] Yanghao Li, Rui Qian, Bowen Pan, Haotian Zhang, Haoshuo Huang, Bowen Zhang, Jialing Tong, Haoxuan You, Xianzhi Du, Zhe Gan, et al. 2025. MANZANO: A Simple and Scalable Unified Multimodal Model with a Hybrid Vision Tokenizer. *arXiv preprint arXiv:2509.16197* (2025).

[22] Junming Lin, Zheng Fang, Chi Chen, Zihao Wan, Fuwen Luo, Peng Li, Yang Liu, and Maosong Sun. 2024. Streamingbench: Assessing the gap for MLLMs to achieve streaming video understanding. *arXiv preprint arXiv:2411.03628* (2024).

[23] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural adaptive video streaming with Pensieve. In *Proceedings of the conference of the ACM special interest group on data communication*. 197–210.

[24] Zili Meng, Xiao Kong, Jing Chen, Bo Wang, Mingwei Xu, Rui Han, Honghao Liu, Venkat Arun, Hongxin Hu, and Xue Wei. 2024. Hairpin: Rethinking packet loss recovery in edge-based interactive video streaming. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*. 907–926.

[25] Zili Meng and Mingwei Xu. 2024. Feedback on Control Path: Early Congestion Feedback. In *Latency Optimization in Interactive Multimedia Streaming*. Springer, 23–42.

[26] Rui Qian, Shuangrui Ding, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. 2025. Dispider: Enabling Video LLMs with Active Real-Time Interaction via Disentangled Perception, Decision, and Reaction. *arXiv preprint arXiv:2501.03218* (2025).

[27] Haoran Qiu, Anish Biswas, Zihan Zhao, Jayashree Mohan, Alind Khare, Esha Choukse, Íñigo Goiri, Zeyu Zhang, Haiying Shen, Chetan Bansal, Ramachandran Ramjee, and Rodrigo Fonseca. 2025. ModServe: Modality- and Stage-Aware Resource Disaggregation for Scalable Multimodal Model Serving. arXiv:2502.00937 [cs.DC] https://arxiv.org/abs/2502.00937

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

[29] Michael Rudow, Francis Y Yan, Abhishek Kumar, Ganesh Ananthanarayanan, Martin Ellis, and KV Rashmi. 2023. Tambur: Efficient loss recovery for videoconferencing via streaming codes. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*. 953–971.

[30] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. 2007. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on circuits and systems for video technology* 17, 9 (2007), 1103–1120.

[31] Chameleon Team. 2024. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818* (2024).

[32] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530* (2024).

[33] V Team, Wenyi Hong, Wenmeng Yu, et al. 2025. GLM-4.5 V and GLM-4.1 V-Thinking: Towards Versatile Multimodal Reasoning with Scalable Reinforcement Learning. *arXiv preprint arXiv:2507.01006* (2025).

[34] Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems* 30 (2017).

[35] Pavan Kumar Anasosalu Vasu, Hadi Pouransari, Fartash Faghri, Raviteja Vemulapalli, and Oncel Tuzel. 2024. Mobileclip: Fast image-text models through multi-modal reinforced training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 15963–15974.

[36] Marko Viitanen, Ari Koivula, Ari Lemmetti, Arttu Ylä-Outinen, Jarno Vanne, and Timo D. Hämäläinen. 2016. Kvazaar: Open-Source HEVC/H.265 Encoder. In *Proceedings of the 24th ACM International Conference on Multimedia* (Amsterdam, The Netherlands). http://doi.acm.org/10.1145/2964284.2973796

[37] Haibo Wang, Bo Feng, Zhengfeng Lai, Mingze Xu, Shiyu Li, Weifeng Ge, Afshin Dehghan, Meng Cao, and Ping Huang. 2025. StreamBridge: Turning Your Offline Video Large Language Model into a Proactive Streaming Assistant. *arXiv preprint arXiv:2505.05467* (2025).

[38] Yuxuan Wang, Yueqian Wang, Bo Chen, Tong Wu, Dongyan Zhao, and Zilong Zheng. 2025. OmniMMI: A Comprehensive Multi-modal Interaction Benchmark in Streaming Video Contexts. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 18925–18935.

[39] Haomiao Xiong, Zongxin Yang, Jiazuo Yu, Yunzhi Zhuge, Lu Zhang, Jiawen Zhu, and Huchuan Lu. 2025. Streaming Video Understanding and Multi-round Interaction with Memory-enhanced Knowledge. *arXiv preprint arXiv:2501.13468* (2025).

[40] Jin Xu, Zhifang Guo, Jinzheng He, Hangrui Hu, Ting He, Shuai Bai, Keqin Chen, Jialin Wang, Yang Fan, Kai Dang, et al. 2025. Qwen2.5-Omni technical report. *arXiv preprint arXiv:2503.20215* (2025).

[41] Francis Y Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. 495–511.

[42] Zhenyu Yang, Yuhang Hu, Zemin Du, Dizhan Xue, Shengsheng Qian, Jiahong Wu, Fan Yang, Weiming Dong, and Changsheng Xu. [n. d.]. SVBench: A Benchmark with Temporal Multi-Turn Dialogues for Streaming Video Understanding. In *The Thirteenth International Conference on Learning Representations*.

[43] Linli Yao, Yicheng Li, Yuancheng Wei, Lei Li, Shuhuai Ren, Yuanxin Liu, Kun Ouyang, Lean Wang, Shicheng Li, Sida Li, et al. 2025. TimeChat-Online: 80% Visual Tokens are Naturally Redundant in Streaming Videos. *arXiv preprint arXiv:2504.17343* (2025).

[44] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, Qianyu Chen, Huarong Zhou, Zhensheng Zou, Haoye Zhang, Shengding Hu, Zhi Zheng, Jie Zhou, Jie Cai, Xu Han, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. MiniCPM-V: A GPT-4V Level MLLM on Your Phone. arXiv:2408.01800 [cs.CV] https://arxiv.org/abs/2408.01800

[45] Lijun Yu, Jose Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. [n. d.]. Language Model Beats Diffusion-Tokenizer is key to visual generation. In *The Twelfth International Conference on Learning Representations*.

[46] Pan Zhang, Xiaoyi Dong, Yuhang Cao, Yuhang Zang, Rui Qian, Xilin Wei, Lin Chen, Yifei Li, Junbo Niu, Shuangrui Ding, et al. 2024. InternLM-XComposer2.5-OmniLive: A comprehensive multimodal system for long-term streaming video and audio interactions. *arXiv preprint arXiv:2412.09596* (2024).

[47] Yuankang Zhao, Qinghua Wu, Gerui Lv, Furong Yang, Jiuhai Zhang, Feng Peng, Yanmei Liu, Zhenyu Li, Ying Chen, Hongyu Guo, et al. 2024. JitBright: towards Low-Latency Mobile Cloud Rendering through Jitter Buffer Optimization. In *Proceedings of the 34th edition of the Workshop on Network and Operating System Support for Digital Audio and Video*. 36–42.

[48] Yiwu Zhong, Zhuoming Liu, Yin Li, and Liwei Wang. 2024. Aim: Adaptive inference of multi-modal LLMs via token merging and pruning. *arXiv preprint arXiv:2412.03248* (2024).