CASA: Cross-Attention over Self-Attention for Efficient Vision-Language Fusion

Moritz Böhle*, Amélie Royer*, Juliette Marrie*, Edouard Grave, and Patrick Pérez

{firstname}@kyutai.org – Kyutai, Paris, France

Abstract. Vision-language models (VLMs) are commonly trained by directly inserting image tokens from a pretrained vision encoder into the text stream of a language model. This allows text and image information to fully attend to one another within the model, but becomes rapidly costly for long multi-image conversations or streaming video applications, both in terms of memory and compute. VLMs leveraging cross-attention are an efficient alternative to token insertion as image tokens are not added to the KV cache. Despite being introduced early on, multimodal cross-attention models are scarce in the current VLM literature and often underperform their token insertion counterparts. In this work, we reinvestigate the effectiveness of cross-attention for vision-language modeling: (i) We analyze the core differences between the cross-attention and self-attention mechanisms, (ii) we train cross-attention VLMs both from a text-only LLM and by adapting a pretrained insertion-based VLM, showing that simple cross-attention is more competitive with token insertion than previously reported, and (iii) we demonstrate the practical advantages of cross-attention on real-time video captioning, where it naturally maintains low latency and near-constant memory cost.

Keywords: multimodality, analysis, streaming video understanding

1 Introduction

Cross-attention (CA) has been employed early on as a lightweight mechanism for fusing multimodal information in transformers [1, 18, 34]. Most recent state-of-the-art VLMs, however, have departed from CA-based fusion, and instead insert visual embeddings into the language model's input stream, directly interleaving them with the text embeddings [4, 8, 36, 46]. While insertion-based fusion has proven remarkably effective, it incurs high computational and memory costs that grow with the number of image tokens, becoming a bottleneck for high-resolution images, multi-image conversations, and streaming video applications.

CA has recently regained interest as a naturally efficient alternative, particularly for streaming applications on long multimodal sequences [6, 21, 31, 44].

* Equal contribution.

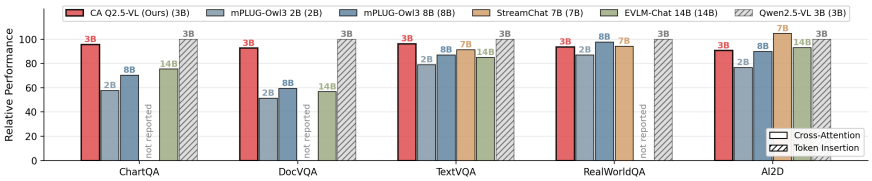


Fig. 1: State-of-the-art cross-attention (CA) VLMs; benchmark scores normalized by Qwen2.5-VL 3B **(hatched grey)**. We adapt Qwen2.5-VL 3B, a token-insertion-based VLM, to use CA. The resulting model **(red)** retains most of the base model's performance and often outperforms prior, larger CA-based VLMs (2B–14B), in particular on high-resolution document understanding tasks (ChartQA, DocVQA).

This has led to multiple architectural variants of CA, in an effort to improve its performance, such as attention gating mechanisms [31, 44], learnable visual tokens [6], or updating visual embeddings across the depth of the network [21]. Nonetheless, CA-based VLMs currently lag behind insertion-based models on several tasks such as document and chart understanding, as shown in Figure 1. The causes of this performance gap remain poorly understood. In particular, it is unclear whether it reflects fundamental limitations of CA or instead arises from differences in training data and implementation choices.

Through this work, we methodically revisit CA as a fusion mechanism. We first analyze the key design differences between CA and token insertion and the efficiency trade-offs they entail, showing that CA offers significant memory and speed advantages at both training and inference time (Tab. 1). To shed light on why current state-of-the-art CA models lag behind recent VLMs, we then directly compare the two mechanisms in two controlled settings: training a CA-based VLM from scratch from a text-only LLM, and adapting a pretrained insertion-based VLM to CA. We find that vanilla CA, without any bells and whistles, performs significantly better than previously reported, narrowing the gap to token insertion to only a few percent on most benchmarks. This simple model outperforms state-of-the-art CA-based VLMs of larger size on most benchmarks, showing the importance of using a comparable modern training pipeline when assessing the performance of CA models with respect to the literature. We further evaluate our CA model on the costly task of real-time video captioning, where CA maintains near-constant memory and latency over long video horizons. In summary, we provide a controlled and comprehensive comparison of CA and token insertion for vision-language fusion. Our contributions are threefold: **(i)** We systematically analyze the core differences between the two mechanisms, identifying five core design elements that progressively bridge CA and token insertion (Sec. 3.2). **(ii)** We train CA VLMs both from a text-only LLM and by adapting a pretrained insertion-based VLM, showing that simple CA is more competitive with token insertion than previously reported. **(iii)** We demonstrate the practical advantages of CA on real-time video captioning [5], where CA enables continuous visual updates with low latency and near-constant memory cost, while token insertion models quickly exhaust their memory budget. Our code and trained models are available at github.com/kyutai-labs/casa.

2 Related work

Insertion-based fusion. Token insertion has become the dominant paradigm for training VLMs: For this, visual embeddings from a pretrained encoder are inserted directly into the language model's input sequence, interleaving them with text embeddings. The visual and textual information thus interact through self-attention layers without any architectural changes. Recent token insertion models [4, 8, 24, 36, 47] have achieved strong multimodal performance with this strategy. However, the number of visual tokens rapidly grows with image resolution or video length, thus increasing the memory and computational costs at both training and inference time. A wide range of techniques has been explored to mitigate this cost, including compressing visual tokens via pixel unshuffling [32], token merging [20], query-based compression [18, 19, 47], or pooling [23, 37], inserting visual tokens in only a subset of layers [7], reducing the cost of attention operations [43], compressing the KV cache at inference [5, 30, 48], or modifying positional embeddings to handle longer contexts [4, 13]. These techniques are largely orthogonal to the choice of fusion mechanism, and could also complement CA to further reduce the number of keys and values in the CA layers.

Cross-attention-based fusion for VLMs was popularized by Flamingo [1], one of the first large-scale VLMs, in which a frozen LLM is conditioned on visual inputs through gated cross-attention. It was adopted in subsequent works [3, 17], and revisited in [6, 21, 44], which leverage the natural scalability of cross-attention for long-context or streaming applications. However, existing cross-attention VLMs have been reported to underperform token-insertion models, in particular on tasks requiring fine-grained visual understanding such as chart or document reading [21, 44]. Prior work has attempted to address this gap by, *e.g.*, updating the visual representations throughout the depth of the model [21], adding register tokens [6], or introducing bespoke architectural changes [44]. We show that much of this gap closes when trained under similar conditions, and cross-attention performs better than previously reported.

Streaming visual understanding. Recently, the VLM literature has seen a gain of interest for streaming video understanding tasks, such as live video captioning or multimodal assistants [5, 41, 48]. Such tasks imply a strict memory and computational budget to execute in real-time and for as long as possible. To address the limitations of token insertion, it is possible to limit the KV cache size through compression [48], pruning of the oldest video frames [5, 30], or by adding condensed "visual memories" to the text stream [41, 50]. In contrast, cross-attention naturally lends itself to efficiently handling a dense stream of images as input. Closest to our work, StreamChat [21] adopts a cross-attention-based design that updates its visual keys and values at each decoding step to align the current text stream with the latest video frame. However, to improve performance of its cross-attention mechanism, StreamChat applies additional dedicated FFN layers to the visual tokens throughout the network. In our ablations, we show that the additional updates of image embeddings, while yielding a small performance boost, incur disproportionate memory and compute cost.

3 Multimodal Fusion: To Cross-Attend or Self-Attend?

Cross-attention's practical benefits stem from the core idea that image tokens never directly enter the transformer's input token stream. Consequently, they are not processed by the language model's self-attention (SA) and feed-forward (FFN) layers and, at inference, are never added to the attention layers' KV cache, making both training and inference more memory- and compute-efficient. However, these strengths may become a limitation when considering the model's performance: Unlike token insertion, image embeddings are not updated throughout the depth of the network, and text tokens cannot directly attend to past images as they are not present in the KV cache. To better understand these distinctions and how they impact the performance-efficiency trade-off, we first formalize the CA and SA mechanisms (Sec. 3.1) and analyze their key differences (Sec. 3.2). Finally, we discuss how to implement CA for scalable multi-image training, allowing us to leverage sequence packing strategies and video inputs (Sec. 3.3).

3.1 Preliminaries

Self-attention in token insertion models. Given text tokens $x = x_{1\ldots T}$ and a tokenized image $y = y_{1\ldots N}$ inserted at position $K < T$ in the text stream, token insertion lets x_T interact with image tokens through self-attention in a standard causal setting as:

$$\mathbf{SA}_{\text{ins}}(x_T|x_{i\leq T}, y) = \text{MHA}(x_T|\, x_{1\ldots K}\, y_{1\ldots N}\, x_{K+1\ldots T})\,; \tag{1}$$

$\text{MHA}(q|k)$ is multi-head attention [35] with query q and k as keys and values.

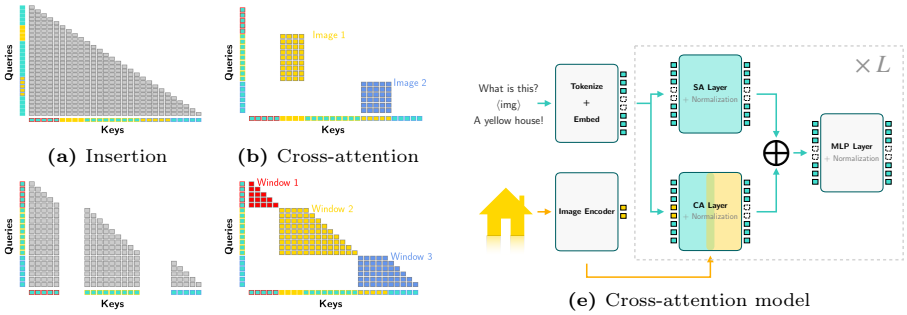
Cross-attention. In contrast, CA injects visual information into text tokens as additive updates. In CA layers, text tokens $x_{i>K}$ which follow the image at timestep K attend to the corresponding image tokens as keys and values:

$$\mathbf{CA}(x_T|y) = \mathbb{1}_{T>K} \times \text{MHA}(x_T|\, y_{1\ldots N})\,. \tag{2}$$

In the remainder of the text, we will refer to the interval $[K, T]$ as the "window" corresponding to the image y seen at time K. Simply speaking, each window is defined as the temporal positions between two input images. Note that in this setting, text tokens $x_{i>K}$ do not attend to past images seen before y by design, and we will further elaborate on this locality property in Sec. 3.2. The output of (2) is then combined, typically via a sum, with the output of the corresponding SA layer in the same transformer block, which operates on text tokens only:

$$\mathbf{SA}_{\text{text}}(x_T|\, x_{i\leq T}) = \text{MHA}(x_T|\, x_{1\ldots T}). \tag{3}$$

In practice, we only consider the scenario where cross-attention layers are placed *in parallel* with self-attention layers in the same transformer block, as illustrated in Fig. 2e: Both CA and SA_{text} operate as separate layers, with their own projections, but on the same inputs. Other designs have also been explored in the literature, such as placing CA after SA [21]. The attention patterns of Eq. (1), Eq. (2), and Eq. (3) are summarized in Fig. 2 (a), (b) and (c) respectively.



(a) Insertion **(b)** Cross-attention **(c)** Text-only SA **(d)** $\text{CA}_{\text{t+v}}$ **(e)** Cross-attention model

Fig. 2: Different attention patterns: (a) full causal self-attention (SA) on image and text, used in standard insertion-based models, **(b)** block-wise cross-attention (CA), where each text query attends to all image tokens in its window, **(c)** text-only causal SA, used alongside CA to allow for text-to-text interaction, and **(d)** $\text{CA}_{\text{t+v}}$, where each text token attends to all preceding text and image tokens in its window (*cf*. Sec. 3.1). In **(e)** we show the general architecture of the CA VLMs we investigate in this work.

3.2 From Cross-Attention to Token Insertion

We identify five core differences between cross-attention and token insertion as fusion mechanisms, each with a different impact on model efficiency and performance. We discuss each of them below, emphasizing how, by combining these core aspects, we can progressively recover self-attention from cross-attention. We summarize these key properties and their impact on model efficiency in Table 1.

(D1) Additional parameters. CA introduces new dedicated layers with additional learnable parameters. In contrast, SA uses the same projection weights to process image and text tokens without distinction. To eliminate this difference, we introduce a parameter-sharing variant of CA, denoted by CA🔗, in which the query, key, value and output projections are shared between the CA and SA layers of a same transformer block. Like token insertion, this variant thus adds no new parameters and we find it to be beneficial for both training and inference speed, as the projection computations can be efficiently merged, see Tab. 1.

(D2) Joint text-image attention and positional embeddings. Inserting images into the token stream allows text tokens to attend to *both* image and other text tokens in the same MHA operation. In addition, both text and image tokens receive temporal positional embeddings (*e.g.*, RoPE). In contrast, in CA layers text tokens only attend to image tokens, and there is traditionally no information on the temporal position of image tokens relative to text tokens. To bridge this gap, we introduce $\text{CA}_{\text{t+v}}$, where text ("t") tokens attend to the last seen visual ("v") tokens *as well as to preceding text tokens* in the same window:

$$\mathbf{CA}_{\text{t+v}}(x_T \mid x_{K<i\leq T}, y) = \mathbb{1}_{T>K} \times \text{MHA}(x_T \mid y_{1\ldots N}\, x_{K+1\ldots T})\,. \tag{4}$$

In other words $\text{CA}_{\text{t+v}}$ can be understood as the SA operation of (1) operating in local windows, rather than across the whole image-text interleaved sequence.

Table 1: Efficiency of cross-attention *vs.* token insertion. CA models can be significantly more efficient than token insertion for training and inference; updating image tokens (D4) and retaining all past images in the KV cache (D5) are the main sources of overhead. We compare cross-attention (CA) and variants with **(D1)**: shared parameters between CA and SA, **(D2)**: text-to-text attention in CA layers, **(D3)**: replacing SA layers with CA, and **(D4.1)**: image updates through FFNs, and token insertion (SA_{ins}). In a single-image setting, SA_{ins} is recovered by adding (D1–5) components to CA. In all settings, we report whether new parameters are required, whether image embeddings evolve throughout the network, as well as memory and throughput measurements for inference and training. Measurements are performed on a single GPU. For inference, we imitate the streaming video captioning setup of Sec. 4.5 in a controlled setting. We report for how many frames per second (FPS) the resulting model would be able to predict 5 text tokens each, while incorporating the new images in a streaming fashion; for details and an extended discussion, see appendix A.

Helium1-2B backbone		Add new params	Update image tokens	Training		Streaming Inference		
				$\text{toks}_{\text{txt}}/\text{s}$	Mem.	FPS	Mem.	#KV
CA		✓	✗	1817	32.9	6.8	6.1	$kT + N$
CA🔗	CA + (D1)	✗	✗	1845	32.0	7.7	5.7	$kT + N$
$\text{CA}_{\text{t+v}}$	CA + (D2)	✓	✗	1560	33.1	5.6	6.1	$kT + (T+N)$
CA🪶	CA + (D3)	✗	✗	1995	29.0	7.6	5.7	kT or N
CA+FFNs	CA + (D4.1)	✓	✓	1504	57.8	6.6	6.1	$kT + N$
SA_{ins}	CA + (D1-5)	✗	✓	1501	62.8	1.2	29.5	$kT + kN$

(D3) Additional layers. Cross-attention operates as an additive residual update to the self-attention output within each transformer block, effectively doubling the number of attention layers. A natural way to reduce this overhead is to instead replace self-attention layers with cross-attention layers in a subset of transformer blocks. We denote this variant by CA🪶. In our experiments, we replace every second SA layer with a CA layer, yielding efficiency gains with only a minor performance drop compared to CA. Note that not all layers can be replaced by CA, as no text-to-text attention would remain. Our $\text{CA}_{\text{t+v}}$ variant is better suited for this setting, as it also computes text-to-text attention, albeit in local windows, and thus constitutes an interesting hybrid between CA and SA.

(D4) Image token updates. With token insertion, the image embeddings are updated through the network in the same way as text tokens: In every transformer block, they pass through a *feed-forward network* (**D4.1**) and attend to preceding tokens via *self-attention* as in Eq. (1) (**D4.2**). In contrast, in cross-attention models image embeddings do not receive any persistent updates, and only undergo different KV projections in every cross-attention layer. This lack of iterative updates may limit the model's ability to further refine image representations and has motivated prior work to enhance CA-based fusion with image updates through dedicated FFNs [21]. Similarly, we denote by CA+FFNs a variant of CA where image embeddings are updated through the FFNs of the network. However, as shown in Tab. 1, this incurs significant memory costs at training and may require sacrifices to fit the memory constraints.

(D5) Multi-image history. As mentioned in Sec. 3.1, CA operates in local windows such that a text token may only attend to the last seen image. In contrast, text tokens in insertion-based models can attend to the whole history, including all past images. This is a key trade-off of CA: Not retaining past images in the KV cache may be detrimental to the model when dealing with multiple images (*e.g.* videos), but also enables lower memory usage at inference, as we will discuss in Sec. 4.5. While prior work has considered integrating multiple past image tokens as keys and values in CA layers [44], we view this as orthogonal and complementary to our study: it would enlarge the set of keys and values that each query token attends to at every decoding step, at the cost of inference speed. Nevertheless, this lack of explicit visual history is not a fundamental limitation: the visual information can instead be carried forward in the text stream. This form of compression has been explored to summarize textual prompts as "gist tokens" [28] and to compress visual tokens within a VLM [45]. In our case, no dedicated tokens are required: since the text tokens that follow an image attend to it through CA, they can encode a compressed representation of it and propagate it to later windows through SA_{text}. This role is naturally filled by the delimiter tokens that, *e.g.*, Qwen's chat template inserts around each image. For consecutive video frames, the post-image delimiters are then the only tokens that carry a frame's information forward. At inference, the model thus accesses the last frame directly through CA and all past frames only through their delimiter tokens. The fact that CA-based VLMs remain competitive on video QA tasks under this compressed visual history (Tab. 4) indicates that the model learns to effectively compress the visual history into these "gist" tokens.

Take-aways. The above discussion reveals that the transition from CA to token insertion can be decomposed into five key design choices (D1-D5). Notably, D1-D3 can be addressed with simple, lightweight modifications to the cross-attention mechanism that can even *reduce* computational overhead. In contrast, D4 (updating image tokens through FFNs) incurs significant memory and compute costs during training, and avoiding D5 (retaining all past images in the KV cache) is a critical design choice for streaming applications, where the accumulation of image tokens introduces substantial memory and latency overhead. This is reflected in the training and inference costs reported in Table 1.

3.3 Scalable Cross-attention Training

We train our models on a mixture of recent public vision-language datasets, primarily composed of question-answer pairs over single images. Due to the high disparity in sequence lengths between samples, we employ multimodal sequence packing, as commonly done in LLM training [10, 42] and modern VLM pipelines [4, 46]. To match the desired cross-attention inference behavior, where text tokens only attend to the image of their corresponding window, we employ the block-wise attention implementation of FlashAttention-2 [9] in the cross-attention layers during training. As illustrated in Fig. 2, we define the attention blocks with the image insertion points acting as natural window delimiters.

4 Experiments

4.1 Experimental Setting

Below we provide a brief summary of our experimental setup. Please refer to Appendix B.1 for full details.

Training Data. We train our models on FineVision [38] and a subset of LLaVA-OneVision-1.5 [2]. Both are curated collections of publicly available image-text datasets covering a wide range of tasks such as captioning, document and chart reading, general VQA, etc. For video training, we further train our models with the aforementioned image-text data alongside LLaVA-Video-178K [51].

Backbones. We investigate both extending language-only models with visual understanding as well as adapting existing VLMs with CA. Specifically, we train our models in two settings: **(i)** Starting from Helium1-2B [16], a text-only LLM, we jointly finetune the backbone and CA layers; **(ii)** We adapt a pretrained VLM, Qwen2.5-VL-3B [4], by learning the additional CA layers while finetuning the LLM backbone at a reduced learning rate. In both scenarios, we use the vision encoder of Qwen2.5-VL [4]. We finetune its last 4 layers when training on image, and freeze it when further finetuning on videos. Finally, we initialize CA layers from the self-attention layers of the respective backbone.

Benchmarks. We evaluate our models on common VLM benchmarks on a variety of tasks: document (DocVQA [27]) and chart (ChartQA [25], InfoVQA [26]) understanding, text recognition (TextVQA [33], OCRBench [22]), and general QA (RealWorldQA [39], AI2D [15], GQA [14], MME [11]). Similarly, we evaluate video models on video understanding (MVBench [19], VideoMME [12], PerceptionTest [29], MLVU [52]), and temporal action reasoning (NExT-QA [40]).

Training Compute. As detailed in Sec. 3.3, we use multimodal sequence packing with block-wise attention during image training, limiting the length of the resulting interleaved text-image sequences to 2048 text tokens and 20,480 image tokens per GPU. We train our models on 64 H100 GPUs with a batch size of 64 and 2 gradient accumulation steps. For token-insertion experiments, we halve the sequence lengths and double the batch size to fit the increased memory cost while processing the same number of tokens as CA models. We process images at native resolution up to $R_{\max}=896^2$ pixels (1024 tokens per image) and downscale larger images, keeping their aspect ratio, to this target resolution. For videos, we use $R_{\max}=504^2$ and extract 2 frames per second, with a maximum clip duration of 3 minutes, resulting in up to 46,080 image tokens per sequence. We train for 40k steps for the Helium1 experiments, 25k steps for Qwen2.5-VL adaptation on images, and an additional 15k steps for training Qwen2.5-VL on videos.

4.2 From LLM to VLM with Cross-Attention

We first train the text-only Helium1-2B with additional CA layers. In Table 2 we compare a Helium1-based VLM trained with token insertion against different CA variants, all trained under the same conditions. For reference, we also report

Table 2: Comparing cross-attention and insertion-based models. Unless specified, all models are built on **2B** LLMs. We use `lmms-eval` [49] for evaluation, and re-evaluate existing models when benchmark results are not provided in the original work. When trained under the same conditions, our CA model reaches performance close to that of a token insertion model, showing the potential of cross-attention as an efficient alternative. Nonetheless, a significant gap remains on infographic understanding, highlighting the benefits of token insertion for certain tasks.

	# params	# train tokens	High-res Doc/Chart			Scene Text		Knowledge / General QA			
			CHART	DOC	INFO	OCRB	TEXT	REALW	AI2D	GQA	MME
Token Insertion – Proprietary											
InternVL2.5 [8]	2.2B	0.5T	79.2	88.7	60.9	804	74.3	60.1	74.9	59.5	2005
Qwen2-VL [36]	2.2B	1.4T	73.5	90.1	65.5	767	79.7	62.9	69.9	59.8	1872
VideoLLaMA3 [46]	2B	-	79.8	91.9	69.4	779	80.1†	67.3	78.2	62.7	1901
Token Insertion – Public data											
SmolVLM [24]	2.2B	-	68.7	80.0	42.2†	729	73.0	51.0†	59.7†	49.2†	1568†
Insert$_{\text{He-2B}}$ (ours)	2.7B	0.13T	79.2	88.3	58.9	746	75.1	60.3	65.5	54.4	1626
Cross-attention SotA – Public data											
mPLUG-Owl3-7B [44]	8B	0.1T	59.2†	55.9†	36.8†	527†	69.0	63.9†	73.4	65.0	1940†
StreamChat [21]	8B	-	◇	◇	◇	◇	72.4	61.7	76.6	62.4	1520
mPLUG-Owl3 2B	2B	0.1T	48.5†	48.2†	28.1†	450†	62.6	56.9†	62.6	61.0	1551†
*Cross-attention (**ours**) – Public data*											
CA	3B	0.13T	77.5	86.4	55.1	740	75.4	58.0	66.0	55.7	1703
CA🔗	2.7B	0.13T	75.2	85.9	51.8	726	74.5	58.0	66.3	55.6	1599
CA🪶	2.7B	0.13T	73.8	84.5	48.0	714	73.0	56.9	64.3	54.0	1607
CA$_{\text{t+v}}$	3B	0.13T	75.3	85.1	52.1	722	74.1	60.3	65.5	54.5	1720

† : Reproduced with the publicly available models at huggingface. ◇: Results and model not publicly available.

performance of proprietary VLMs (InternVL2.5 [8], Qwen2-VL [36], and VideoLLaMA3 [46]), an open-source VLM trained on public data (SmolVLM [24]) and recent CA-based VLMs (mPLUG-Owl3 [44], StreamChat [21]).

As shown in Table 2, the performance of the vanilla cross-attention model, as well as its variants, is very close to the token insertion model trained in the same setting, with an average 1.5 percent drop of performance. A significant gap only remains on InfographicVQA, which requires understanding complex, high-resolution figures. Compared to the literature, our CA models outperform the CA-based mPLUG-Owl3 on most benchmarks, even surpassing its 7B variant, highlighting the importance of an up-to-date training pipeline to fairly compare CA models with token insertion. When comparing different variants of cross-attention, we first note that including text tokens among the keys and values of the CA layers (CA$_{\text{t+v}}$) performs similarly to the vanilla CA on average. Second, we observe that sharing CA and SA parameters (CA🔗) is detrimental to the model performance, but only by a small margin. This introduces an interesting efficiency-performance trade-off, as CA🔗 is more efficient than CA, as shown in Table 1. Finally, replacing every other self-attention layer with CA layers (CA🪶) further reduces compute while incurring only a small drop in performance. The robustness of CA across these variants is encouraging from a practical standpoint: at only a minor performance cost in visual understanding tasks, the most efficient CA versions (CA🔗, CA🪶) can process over 6× more frames per second

Table 3: Adapting a pretrained Qwen2.5-VL by training additional CA layers (and last 4 image encoder blocks) for 25k training steps. Training only the CA layers (second row) already recovers most of the performance of the base model at a low training cost, and can be further improved by finetuning the backbone alongside CA layers (first row). Notably, a gap still remains on complex chart or document understanding tasks such as InfoVQA. In Table 6, we show how updating image embeddings through FFNS in CA (D4) can further reduce this gap, but does not entirely close it.

	Document/Chart			Scene Text		Knowledge / General QA			
	Chart	Doc	Info	OcrB	Text	RealW	AI2D	GQA	MME
Qwen2.5-VL 3B	84.0	93.0	77.1	797	79.3	65.4	81.6	–	2157
	83.1†	92.4†	75.1†	796†	79.6†	60.4†	79.6†	61.0†	2224†
CA	81.8	87.9	58.2	796	76.5	62.1	76.1	59.4	1979
CA (frozen Qwen)	80.9	86.8	58.1	775	77.0	58.3	75.3	58.8	1854
CA🔗	81.4	87.8	59.1	795	76.6	63.0	74.4	59.2	2025
CA$_{t+v}$	81.3	87.1	56.8	775	76.3	63.8	74.4	58.6	1971

† : Reproduced with the publicly available model at huggingface.

and use over 5× less memory at streaming video inference than token insertion (see Tab. 1), which we further discuss in Sec. 4.5.

4.3 Adapting a VLM from Insertion to Cross-Attention

The previous results show that CA is competitive with token insertion when both are trained from scratch. In practice, however, adapting strong pretrained VLMs would be significantly more efficient. We therefore investigate whether a pretrained insertion-based model can be efficiently converted to cross-attention. Specifically, we adapt Qwen2.5-VL-3B [4] by replacing its token-insertion mechanism with CA layers. We train the CA layers and the last four blocks of the visual encoder, while finetuning the LLM backbone with a smaller learning rate.

Image evaluation. We report results in Table 3, directly comparing to the base model Qwen2.5-VL. As in Sec. 4.2, changing to a CA-based design incurs a moderate performance drop (6.8% on average). Consistent with the trends observed in the previous section, the most significant gap occurs on InfographicVQA. Importantly, this conversion requires only 25k training steps, after which the resulting CA model recovers most of the base model's capabilities while also gaining the practical efficiency advantages of cross-attention at inference time.

Video evaluation. We further finetune our Qwen-CA model on videos [51] and report results in Table 4. As discussed in Section 3.2 (D5), we implement cross-attention with windows, such that each text token only attends to the most recent video frame. Thus, to preserve information from past images in the text stream, we rely on the post-image delimiter tokens, already inserted by Qwen-VL's chat template after each frame, to act as gist tokens. In particular, this differs from past CA works, which generally insert multiple past frames as key-value inputs in cross-attention layers, thus increasing the cost of the CA operation. Despite this limited visual history, our CA model lands only

Table 4: Video benchmarks. Adapting Qwen2.5-VL with CA layers retains much of its performance and is on par with the larger cross-attention-based mPLUG-Owl3-7B. Results are reported for CA_{t+v} either in the standard evaluation setting with a single window (✗) or in a one-frame-per-window setting (✓), where past visual information is carried only through image delimiter tokens (see discussion on 'gist' tokens, Sec. 3.2).

	Max #Frames	Windows	VideoMME w/o sub	VideoMME w/ sub	Next QA	Percep. Test	MV Bench	MLVU
Qwen2.5VL 3B	768	✗	61.5	67.6	-	66.9	67.0	68.2
	360		58.8†	63.6†	78.9†	67.0†	66.2†	65.2†
StreamChat 7B	40	✗	58.6	62.8	78.5	◇	53.3	63.9
mPLUG-Owl3 7B	128	✗	53.5	-	78.6	-	54.5	-
mPLUG-Owl3 7B†	180	✗	59.8†	65.2†	82.1†	65.6†	58.3†	65.6†
CA_{t+v}	180	✗	59.0	64.8	79.8	63.8	60.6	67.9
CA_{t+v}	180	✓	56.9	60.5	78.8	62.6	59.5	66.2

† : Reproduced with the publicly available model at huggingface.

3.9 percent below the base model on average, suggesting that it effectively stores relevant visual information in the gist tokens. For reference, we also train a model in the standard evaluation setting, where all image frames are inserted in a single window (second-to-last row). This leads to a performance boost (but a higher compute cost) and performs on-par or better than prior CA works of larger size.

4.4 Ablation Experiments

For ease of reading, we group the 9 benchmarks into 3 categories in our ablation results: HRES, high-res. chart and document reading (DocVQA, InfoVQA, ChartQA), OCR, reading text in natural images (OCRBench, TextVQA), and VQA, general-knowledge visual understanding (RealWorldQA, AI2D, GQA, MME).

Cross-attention layer frequency. In Table 5 (left), we report results for training CA🔥 from the text-only Helium1-2B model [16] where CA layers *replace* SA every n layers, with $n = 2$ and 4. Note that the CA🔥 results of Table 2 use $n = 2$ by default. Reducing the number of CA layers (here, replacing SA every 4 layers instead of every 2) trades a small amount of performance for additional efficiency, providing a simple way to adapt to different compute budgets.

Updating image tokens. Recent CA-based VLMs [21] update the image embeddings by applying dedicated FFN layers to improve performance. We evaluate the benefits of this approach through a small-scale ablation with the image encoder frozen, as propagating the image tokens through FFNs substantially increases memory usage (*cf*. Table 1). We compare updating image tokens through FFNs ($CA_{Q2.5\text{-}VL}$ + FFN Updates) against our $CA_{Q2.5\text{-}VL}$ baseline and the original Qwen2.5-VL model. We also include Qwen2.5-VL *finetuned* on our data distribution ($Insert_{Qw}$, ft), which controls for the data-distribution gap between the pretrained Qwen2.5-VL model and our CA variants, allowing a fairer comparison of the underlying modeling choices. As shown in Table 6, updating image tokens

Table 5: CA-layer frequency and token compression. (a) Effect of inserting CA at different layer intervals, in the same training setting as Table 2; using fewer CA layers trades a small drop in performance for additional efficiency. **(b)** Impact of token compression, a common approach to improve the efficiency of token insertion. Reducing the number of image tokens quickly deteriorates performance on tasks involving high resolution images. Furthermore, as we show in Section 4.5, token compression is not sufficient to solve the memory bottleneck inherent to image token insertion, as the KV cache still grows during streaming inference.

	(a) CA$^{\text{l+v}}$ layer frequency			*(b) Impact of token compression*		
Task	CA	CA (every n layers) $n=2$	$n=4$	Insertion (1156 tokens)	QFormer (128)	QFormer (32)
hres	70.8	68.4	67.8	**74.2**	62.7	56.0
ocr	73.1	72.0	71.1	**74.5**	70.8	67.0
vqa	60.4	58.8	57.0	**60.6**	57.8	56.9
avg.	68.1	66.4	65.3	**69.8**	63.8	59.9

Table 6: Updating image tokens. Propagating image tokens through the underlying LLM's FFNs improves performance, but incurs non-negligible memory and compute costs. In addition, it doesn't fully close the gap to token insertion on complex text reading tasks such as InfoVQA, highlighting that token insertion still holds benefit on particular tasks. Note that all models in this table are trained under identical conditions, but they use roughly 3x fewer tokens than the Qwen2.5-VL setting reported in the main table, as ablations were conducted under a reduced compute budget.

Model	AI2D	ChartQA	DocVQA	InfoVQA
Qwen2.5-VL	79.6	83.1	92.4	75.1
$\text{Insert}_{\text{Qw}}$ (ft)	78.9	83.8	91.3	68.8
CA + FFN Update	76.2	80.6	86.9	59.2
CA	73.7	78.3	83.1	52.1
CA	71.8	78.1	82.9	51.7

improves performance ($\approx$ 4pp on reading tasks), in line with prior work [21], but comes at significant additional memory costs during training. The comparison to Qwen2.5-VL finetuned on our data ($\text{Insert}_{\text{Qw}}$, ft) further suggests that part of the gap to Qwen2.5-VL stems from its stronger, partly closed-source training data rather than the fusion mechanism: on InfoVQA, this finetuning alone lowers the score from 75.1 to 68.8, accounting for 6.3pp of the gap to our CA variants.

Link to token compression. To reduce token insertion costs, it is common to compress the number of image tokens before inserting them in the text stream [18, 20, 23, 37, 47]. In Table 5 (right), we report results of training Helium1-2B with either full token insertion or a Q-Former-based compression [18], in which a small transformer block is applied to the N image tokens produced by the vision encoder, alongside $Q \ll N$ learnable queries; only the Q queries

are then inserted in the LLM's textual stream. For general VQA tasks, we find that even aggressive token compression has limited impact on the performance, but for tasks requiring more detailed representations (HRES, OCR) we observe significant performance drops. Hence, compressed insertion can be a practical alternative to full token insertion if the task does not involve fine-grained visual detail. Nonetheless, as we discuss in Appendix D, even with token compression the memory cost and context length limit quickly become a bottleneck when dealing with long streaming video understanding. Finally, note that token compression is orthogonal and can be combined with CA to further reduce the number of tokens in the CA layers.

This video shows [00:01|0.0s] *the Apollo 13* [00:02|0.0s] *spacecraft during its* [00:03|0.0s] *40 years of space camp missions at* [00:05|0.0s] *NASA's Johnson Space Center* [00:06|0.0s] *in Houston Texas* [00:07|0.0s] *and it was filmed by NASA* [00:08|0.0s] *Space Camp Director* [00:09|0.0s] *Steve Smith who has been with us for over* [00:10|0.0s] *25 years now* [00:11|0.0s] *at this point he had to go back home* [00:12|0.0s] *to his job as a* [00:13|0.0s] *director but we're going* [00:15|0.0s] *with him here today* [00:16|0.0s] *so you can see how they've changed* [00:17|0.0s] *over time from* [00:18|0.0s] *Apollo era* [00:19|0.0s] *missions all right* [00:20|0.0s] *we'll start*

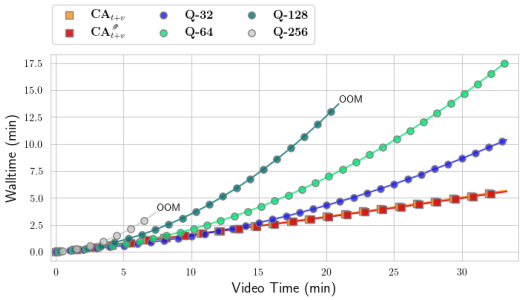
Fig. 3: Live captioning. We display captions generated by our CA_{t+v}-Qwen2.5VL model. Each text span is annotated with the corresponding frame's timestamp (*top*) and the model's delay as [timestamp | delay]. As shown in Figure 4b, the model's outputs are generated much faster than real-time with no noticeable increase over time and near-constant memory usage, as only new text tokens are added to the model's KV cache; for further qualitative examples, including insertion-based VLMs, see App. D.

4.5 Application to Live Video Captioning

An important motivation for efficient image-text fusion are streaming video understanding applications. This comes with two particular challenges: **(i)** Memory growth needs to be controlled in order to handle longer videos, and **(ii)** for streaming applications, the model latency needs to stay below the frame rate to avoid accumulating delay over time. By design, CA is well equipped to address both of these issues. To illustrate this, we consider the task of live video captioning. Specifically, we finetune our CA_{t+v}-adapted Qwen2.5-VL model trained on images only on Live-WhisperX-526K, a recent instruction-tuning dataset [5]. It is composed of video frames sampled at 2fps and interleaved with the text transcript of the video's original audio. As discussed in Sec. 3.2, we design our CA layers such that text tokens only attend to the latest image in the current window. At inference, this means we simply continuously execute the model while

Model	Training steps	Winrate (%)
CA_{t+v} (Qwen2.5-VL) (Ours, 3B)		
	3k	25.4
	6k	27.6
	8k	33.7
	12k	34.3
	15k	36.3
	17k	39.4
	20k	39.0
Baselines as reported in LiveCC, 7B		
LiveCC-7B-Base		43.2
LiveCC-7B-Instruct		41.5
Qwen2-VL-7B-LiveCC		33.7

(a) LLM-as-judge evaluation on LiveSports video captioning



(b) Wall-clock time measurements in the live captioning setting for CA and token-insertion models

Fig. 4: Quantitative results on LiveCC. (a) We evaluate our CA_{t+v} model on the LiveSports captioning tasks proposed in [5], following their LLM-as-a-judge methodology. **(b)** We record the wall-time as a function of the number of frames inserted in a streaming captioning scenario, for CA and token-compression techniques (Q-Former with different numbers of query tokens). While token compression mitigates the computational cost of token insertion for short videos, the CA model maintains low latency inference for longer times. In addition, token insertion tends to run out of memory quickly, especially for larger numbers of tokens. Note that for better readability, we only plot a subset of markers, although the plotted measurements occur at every frame.

replacing the key-value sources of the CA layers every half a second. We provide qualitative samples of live captioning results in Figure 3 and Appendix D.

Quantitative evaluation. We evaluate our LiveCC-tuned model on the LiveSports3K benchmark proposed in [5]. The dataset consists of videos of sports events ($\sim$ 20 seconds long). The captions are evaluated using an LLM as a judge, following the evaluation protocol provided in LiveCC's repository with GPT-4o acting as the judge. In Figure 4(a), we report results for a CA_{t+v} Qwen2.5-VL-3B model trained on LiveCC and evaluated across training steps, and compare to the results reported in the original LiveCC paper. Notably, despite its smaller size (3B *vs*. 7B), our CA model obtains scores close to those of LiveCC.

Real-world performance. As can be seen in Figure 4(b), the memory cost of token insertion methods increases more rapidly than for the CA-based model. While token compression reduces the cost of token insertion for short conversations, it cannot alone prevent the increased memory usage leading to OOM when the number of tokens is too high. In Figure 4, we also report the wall-time of the same models (recorded on a single H100 GPU) as a function of the number of frames inserted. As expected, generation with insertion-based models becomes progressively slower as image tokens accumulate in the KV cache. In contrast, the CA-based model maintains high inference speed over much longer horizons. Together, these results show that the efficiency properties of CA translate into tangible advantages for real-world streaming, enabling low-latency captioning over extended time horizons where insertion-based models are impractical.

5 Conclusions

We revisited CA as a fusion mechanism for VLMs, a design that has been largely set aside in favor of token insertion despite CA's favorable efficiency properties.

We identified five core design differences (D1-D5) that progressively bridge CA and token insertion, and analyzed their respective impact on efficiency and performance. Training CA-based VLMs both from a text-only LLM and by adapting a pretrained insertion-based model, we found that simple cross-attention, without any of the elaborate modifications proposed in prior work, performs close to token insertion on most image and video benchmarks.

Notably, our experiments on adapting a pretrained insertion-based VLM to cross-attention yield, to the best of our knowledge, the strongest CA-based models to date, outperforming prior CA-based VLMs of significantly larger size on many benchmarks (Fig. 1). This adapted model naturally lends itself to streaming applications: by simply replacing the key-value sources of the CA layers at each new frame, it performs live video captioning with near-constant memory and latency, while token insertion models quickly exhaust their memory budget. This efficiency advantage makes CA especially attractive for real-time video understanding and long multi-image interactions, while token insertion may remain preferable for fine-grained document or infographics understanding and long-range retrieval when latency and memory are less of a constraint.

Our results suggest that cross-attention deserves renewed consideration as a practical and competitive alternative for vision-language fusion, especially as applications move toward longer, streaming multimodal inputs.

Acknowledgements. This project is funded by Iliad Group, CMA CGM Group and Schmidt Sciences. The authors thank Alexandre Défossez for his support and feedback throughout the project.

Bibliography

[1] Alayrac, J.B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al.: Flamingo: a Visual Language Model for Few-Shot Learning. Adv. Neural Inform. Process. Syst. (2022)

[2] An, X., Xie, Y., Yang, K., Zhang, W., Zhao, X., Cheng, Z., Wang, Y., Xu, S., Chen, C., Wu, C., et al.: Llava-onevision-1.5: Fully open framework for democratized multimodal training. arXiv:2509.23661 (2025)

[3] Awadalla, A., Gao, I., Gardner, J., Hessel, J., Hanafy, Y., Zhu, W., Marathe, K., Bitton, Y., Gadre, S., Sagawa, S., Jitsev, J., Kornblith, S., Koh, P.W., Ilharco, G., Wortsman, M., Schmidt, L.: OpenFlamingo: An Open-Source Framework for Training Large Autoregressive Vision-Language Models. arXiv:2308.01390 (2023)

[4] Bai, S., Chen, K., Liu, X., Wang, J., Ge, W., Song, S., Dang, K., Wang, P., Wang, S., Tang, J., et al.: Qwen2.5-VL Technical Report. arXiv:2502.13923 (2025)

[5] Chen, J., Zeng, Z., Lin, Y., Li, W., Ma, Z., Shou, M.Z.: LiveCC: Learning Video LLM with Streaming Speech Transcription at Scale. In: IEEE Conf. Comput. Vis. Pattern Recog. (2025)

[6] Chen, K., Shen, D., Zhong, H., Zhong, H., Xia, K., Xu, D., Yuan, W., Hu, Y., Wen, B., Zhang, T., et al.: EVLM: An Efficient Vision-Language Model for Visual Understanding. arXiv:2407.14177 (2024)

[7] Chen, L., Zhao, H., Liu, T., Bai, S., Lin, J., Zhou, C., Chang, B.: An Image is Worth 1/2 Tokens After Layer 2: Plug-and-Play Inference Acceleration for Large Vision-Language Models. In: Eur. Conf. Comput. Vis. (2024)

[8] Chen, Z., Wang, W., Cao, Y., Liu, Y., Gao, Z., Cui, E., Zhu, J., Ye, S., Tian, H., Liu, Z., et al.: Expanding Performance Boundaries of Open-Source Multimodal Models with Model, Data, and Test-Time Scaling. arXiv:2412.05271 (2024)

[9] Dao, T.: FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In: Int. Conf. Learn. Represent. (2024)

[10] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al.: The llama 3 herd of models. 2407.21783 (2024)

[11] Fu, C., Chen, P., Shen, Y., Qin, Y., Zhang, M., Lin, X., Qiu, Z., Lin, W., Yang, J., Zheng, X., Li, K., Sun, X., Ji, R.: Mme: A comprehensive evaluation benchmark for multimodal large language models. ArXiv **abs/2306.13394** (2023), `https://api.semanticscholar.org/CorpusID:259243928`

[12] Fu, C., Dai, Y., Luo, Y., Li, L., Ren, S., Zhang, R., Wang, Z., Zhou, C., Shen, Y., Zhang, M., et al.: Video-MME: The First-Ever Comprehensive Evaluation Benchmark of Multi-modal LLMs in Video Analysis. In: IEEE Conf. Comput. Vis. Pattern Recog. (2025)

[13] Ge, J., Chen, Z., Lin, J., Zhu, J., Liu, X., Dai, J., Zhu, X.: V2pe: Improving multimodal long-context capability of vision-language models with variable visual position encoding. ArXiv **abs/2412.09616** (2024), https://api.semanticscholar.org/CorpusID:274656197
[14] Hudson, D.A., Manning, C.D.: GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering. In: IEEE Conf. Comput. Vis. Pattern Recog. (2019)
[15] Kembhavi, A., Salvato, M., Kolve, E., Seo, M., Hajishirzi, H., Farhadi, A.: A Diagram Is Worth A Dozen Images. In: Eur. Conf. Comput. Vis. (2016)
[16] Kyutai: Helium1: A modular and multilingual LLM (2025), https://huggingface.co/kyutai/helium-1-2b
[17] Li, B., Zhang, Y., Chen, L., Wang, J., Pu, F., Cahyono, J.A., Yang, J., Li, C., Liu, Z.: Otter: A Multi-Modal Model with In-Context Instruction Tuning. IEEE Trans. Pattern Anal. Mach. Intell. (2025)
[18] Li, J., Li, D., Savarese, S., Hoi, S.: BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. In: International Conference on Machine Learning (ICML) (2023)
[19] Li, K., Wang, Y., He, Y., Li, Y., Wang, Y., Liu, Y., Wang, Z., Xu, J., Chen, G., Luo, P., et al.: MVBench: A Comprehensive Multi-modal Video Understanding Benchmark. In: IEEE Conf. Comput. Vis. Pattern Recog. (2024)
[20] Li, X., Wang, Y., Yu, J., Zeng, X., Zhu, Y., Huang, H., Gao, J., Li, K., He, Y., Wang, C., Qiao, Y., Wang, Y., Wang, L.: VideoChat-Flash: Hierarchical Compression for Long-Context Video Modeling. arXiv:2501.00574 (2024)
[21] Liu, J., Yu, Z., Lan, S., Wang, S., Fang, R., Kautz, J., Li, H., Alvare, J.M.: StreamChat: Chatting with Streaming Video. arXiv:2412.08646 (2024)
[22] Liu, Y., Li, Z., Huang, M., Yang, B., Yu, W., Li, C., Yin, X.C., Liu, C.L., Jin, L., Bai, X.: OCRBench: On the Hidden Mystery of OCR in Large Multimodal Models. Science China Information Sciences **67**(12) (2024)
[23] Maaz, M., Rasheed, H., Khan, S., Khan, F.S.: Video-ChatGPT: Towards Detailed Video Understanding via Large Vision and Language VideoLLaMA 3: Frontier Multimodal Foundation Models for Image and Video Understanding . In: Proceedings of the Association for Computational Linguistics (ACL) (2024)
[24] Marafioti, A., Zohar, O., Farré, M., Noyan, M., Bakouch, E., Cuenca, P., Zakka, C., Allal, L.B., Lozhkov, A., Tazi, N., et al.: SmolVLM: Redefining small and efficient multimodal models. arXiv:2504.05299 (2025)
[25] Masry, A., Do, X.L., Tan, J.Q., Joty, S., Hoque, E.: ChartQA: A Benchmark for Question Answering about Charts with Visual and Logical Reasoning. In: Findings of the association for computational linguistics: ACL (2022)
[26] Mathew, M., Bagal, V., Tito, R., Karatzas, D., Valveny, E., Jawahar, C.: InfographicVQA. In: WACV (2022)
[27] Mathew, M., Karatzas, D., Jawahar, C.: DocVQA: A Dataset for VQA on Document Images. In: WACV (2021)
[28] Mu, J., Li, X.L., Goodman, N.D.: Learning to compress prompts with gist tokens. ArXiv **abs/2304.08467** (2023), https://api.semanticscholar.org/CorpusID:258179012

[29] Patraucean, V., Smaira, L., Gupta, A., Recasens, A., Markeeva, L., Banarse, D., Koppula, S., Malinowski, M., Yang, Y., Doersch, C., et al.: Perception Test: A Diagnostic Benchmark for Multimodal Video Models. In: Adv. Neural Inform. Process. Syst. (2023)
[30] Qian, R., Dong, X., Zhang, P., Zang, Y., Ding, S., Lin, D., Wang, J.: Streaming Long Video Understanding with Large Language Models. In: Adv. Neural Inform. Process. Syst. (2024)
[31] Royer, A., Böhle, M., de Marmiesse, G., Mazaré, L., Zeghidour, N., Défossez, A., Pérez, P.: Vision-Speech Models: Teaching Speech Models to Converse about Images. arXiv:2503.15633 (2025)
[32] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A.P., Bishop, R., Rueckert, D., Wang, Z.: Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In: IEEE Conf. Comput. Vis. Pattern Recog. (2016)
[33] Singh, A., Natarjan, V., Shah, M., Jiang, Y., Chen, X., Parikh, D., Rohrbach, M.: Towards VQA Models That Can Read. In: IEEE Conf. Comput. Vis. Pattern Recog. (2019)
[34] Tan, H.H., Bansal, M.: Lxmert: Learning cross-modality encoder representations from transformers. In: Conference on Empirical Methods in Natural Language Processing (2019), `https://api.semanticscholar.org/CorpusID:201103729`
[35] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is All you Need. In: Adv. Neural Inform. Process. Syst. (2017)
[36] Wang, P., Bai, S., Tan, S., Wang, S., Fan, Z., Bai, J., Chen, K., Liu, X., Wang, J., Ge, W., et al.: Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution. arXiv:2409.12191 (2024)
[37] Wang, Y., Li, X., Yan, Z., He, Y., Yu, J., Zeng, X., Wang, C., Ma, C., Huang, H., Gao, J., et al.: InternVideo2.5: Empowering Video MLLMs with Long and Rich Context Modeling. arXiv:2501.12386 (2025)
[38] Wiedmann, L., Zohar, O., Mahla, A., Wang, X., Li, R., Frere, T., von Werra, L., Gosthipaty, A.R., Marafioti, A.: FineVision: Open Data Is All You Need (2025)
[39] xAI: Grok-1.5 vision preview (Apr 2024), dataset obtained at `https://huggingface.co/datasets/lmms-lab/RealWorldQA`
[40] Xiao, J., Shang, X., Yao, A., Chua, T.S.: Next-qa: Next phase of question-answering to explaining temporal actions. In: IEEE Conf. Comput. Vis. Pattern Recog. (2021)
[41] Xu, R., Xiao, G., Chen, Y., He, L., Peng, K., Lu, Y., Han, S.: StreamingVLM: Real-Time Understanding for Infinite Video Streams. arXiv:2510.09608 (2025)
[42] Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., et al.: Qwen3 technical report. arXiv:2505.09388 (2025)
[43] Yang, S., Wang, B., Zhang, Y., Shen, Y., Kim, Y.: Parallelizing linear transformers with the delta rule over sequence length. ArXiv **abs/2406.06484** (2024), `https://api.semanticscholar.org/CorpusID:270371554`

[44] Ye, J., Xu, H., Liu, H., Hu, A., Yan, M., Qian, Q., Zhang, J., Huang, F., Zhou, J.: mPLUG-Owl3: Towards Long Image-Sequence Understanding in Multi-Modal Large Language Models. In: Int. Conf. Learn. Represent. (2025)
[45] Ye, X., Gan, Y., Huang, X., Ge, Y., Tang, Y.: VoCo-LLaMA: Towards Vision Compression with Large Language Models. In: IEEE Conf. Comput. Vis. Pattern Recog. (2025)
[46] Zhang, B., Li, K., Cheng, Z., Hu, Z., Yuan, Y., Chen, G., Leng, S., Jiang, Y., Zhang, H., Li, X., et al.: VideoLLaMA 3: Frontier Multimodal Foundation Models for Image and Video Understanding. arXiv:2501.13106 (2025)
[47] Zhang, H., Li, X., Bing, L.: Video-LLaMA: An Instruction-tuned Audio-Visual Language Model for Video Understanding. arXiv:2306.02858 (2023)
[48] Zhang, H., Wang, Y., Tang, Y., Liu, Y., Feng, J., Jin, X.: Flash-VStream: Efficient Real-Time Understanding for Long Video Streams. In: Int. Conf. Comput. Vis. (2025)
[49] Zhang, K., Li, B., Zhang, P., Pu, F., Cahyono, J.A., Hu, K., Liu, S., Zhang, Y., Yang, J., Li, C., et al.: Lmms-eval: Reality check on the evaluation of large multimodal models. In: Findings of the Association for Computational Linguistics: NAACL 2025. pp. 881–916 (2025)
[50] Zhang, P., Dong, X., Cao, Y., Zang, Y., Qian, R., Wei, X., Chen, L., Li, Y., Niu, J., Ding, S., et al.: InternLM-XComposer2.5-OmniLive: A Comprehensive Multimodal System for Long-term Streaming Video and Audio Interactions. arXiv:2412.09596 (2024)
[51] Zhang, Y., Wu, J., Li, W., Li, B., Ma, Z., Liu, Z., Li, C.: Video Instruction Tuning With Synthetic Data (2024), arXiv:2410.02713
[52] Zhou, J., Shu, Y., Zhao, B., Wu, B., Liang, Z., Xiao, S., Qin, M., Yang, X., Xiong, Y., Zhang, B., et al.: MLVU: Benchmarking Multi-task Long Video Understanding. In: Proceedings of the Computer Vision and Pattern Recognition Conference. pp. 13691–13701 (2025)