Codec Primitives For Efficient Video Language Models

Sayan Deb Sarkar

Stanford University

Google Deepmind

April 2, 2026

Who am I?

PhD student since 2024.09

- Advisor: Prof. Iro Armeni
- Gradient Spaces Research Group, part of Stanford Vision and Learning Lab (SVL)

Computer Science MSc 2022.09 – 2024.08

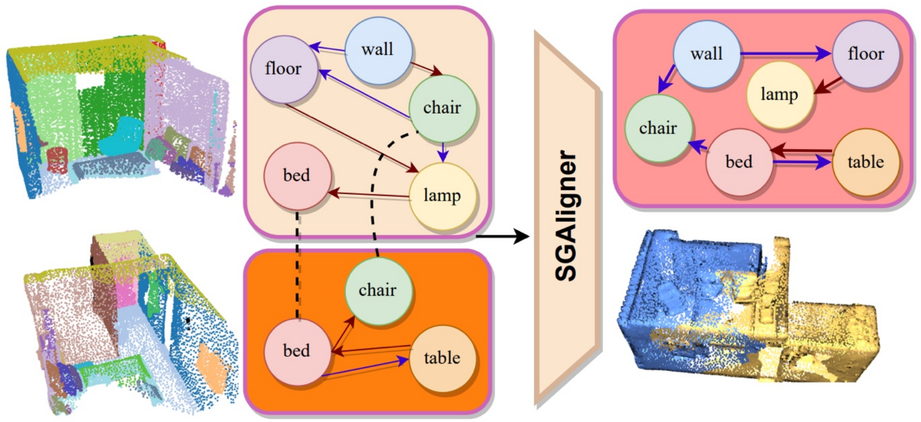
- Advisor: Prof. Marc Pollefeys
- Computer Vision And Geometry Group (CVG)

At Industry

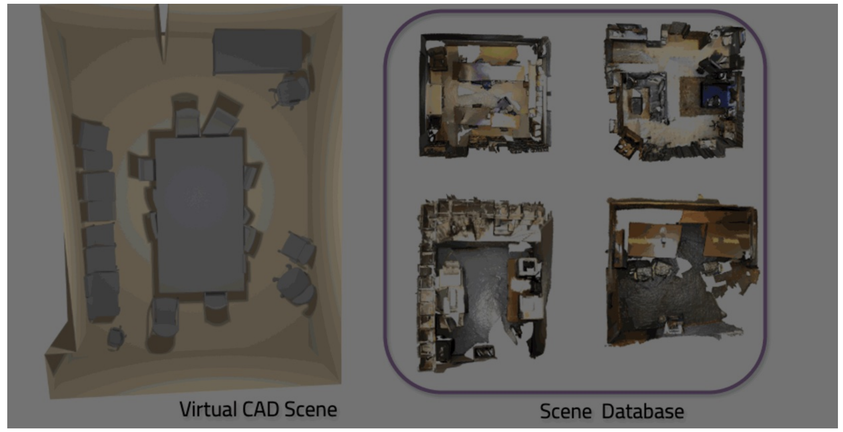
- Incoming intern at Waymo Perception Research (Summer ‘26)
- Internships at Microsoft Spatial AI Lab (PhD, Summer ‘25) & Qualcomm XR (MS)
- Computer Vision Engineer at Mercedes Benz R & D

sayands.github.io

My Work Until Now

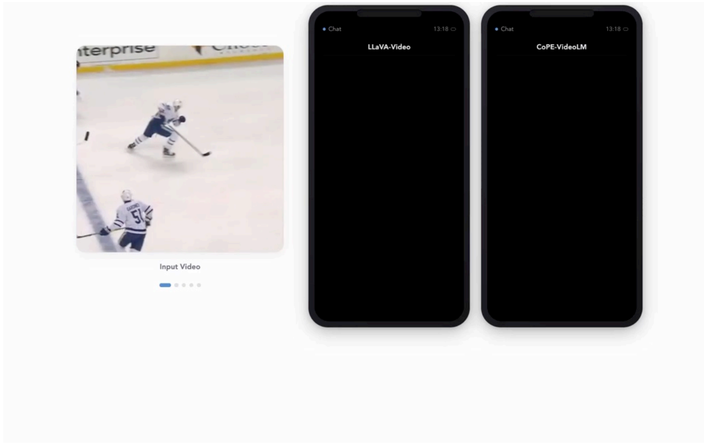


SGAligner [ICCV 2023]



CrossOver [CVPR 2025 Highlight]

GuideFlow3D [NeurIPS 2025]



CoPE-VideoLM [arXiv 2026]

Theme: Multimodal Understanding and Spatial Intelligence

Microsoft Stanford University ETH zürich

CoPE-VideoLM

Leveraging Codec Primitives For Efficient Video Language Modeling

Sayan Deb Sarkar° Rémi Pautrat Ondrej Miksik Marc Pollefeys

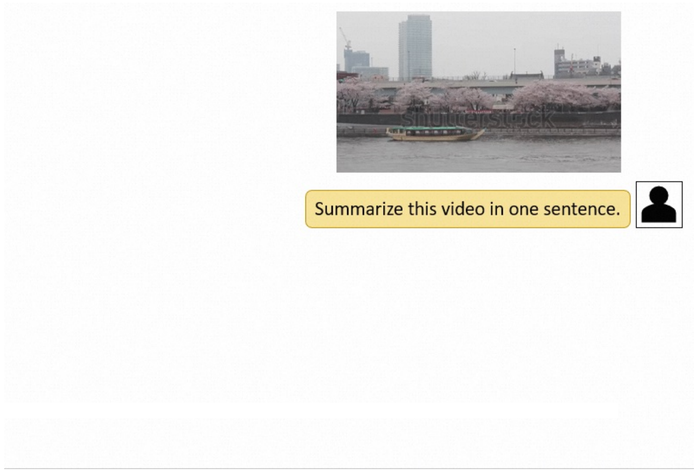
Iro Armeni Mahdi Rad* Mihai Dusmanu*

° Part of work done at Microsoft

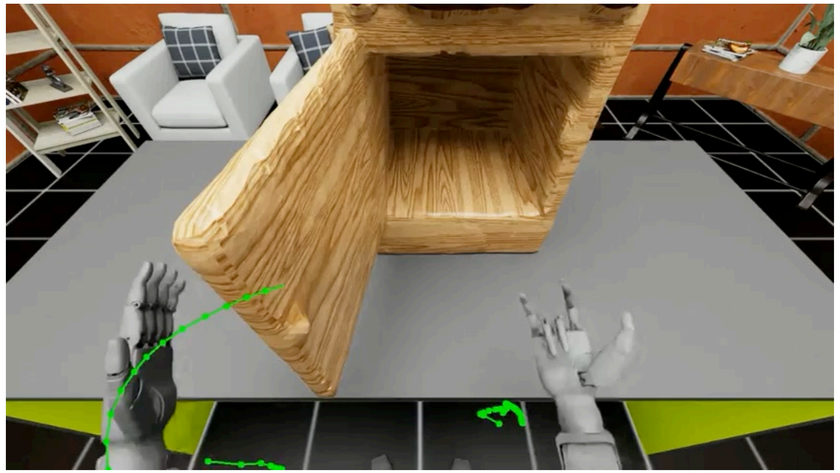
* Equal Supervision

Why Video Language Models?

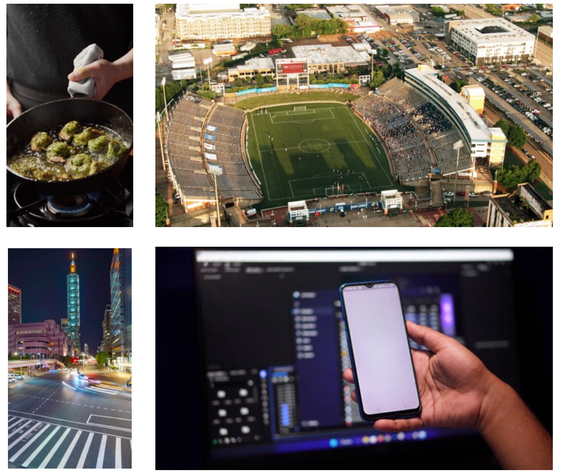
From question-answering to robotics, only images are not enough!



Video Question Answering



Robotics and Embodied Agents



Video Retrieval

Goal: Build models that reason about temporal dynamics, not just static snapshots.

How do current VideoLMs process video?

Sample keyframes → encode full image → feed all tokens to the LLM

Sample fixed number of frames regardless of video length

Question: Who pushed player number 52?

Vision Encoder

Tokenizer

Large Language Model

Output: Player number 46 from the Boston Bruins pushed player number 52.

Problem: Every frame is fully decoded and tokenized with the same budget, ignoring inherent video sparsity.

The Cost of Brute-Force Encoding

Sparse sampling misses events, while dense encoding wastes tokens on redundancy

What gets lost in sparse keyframe sampling?

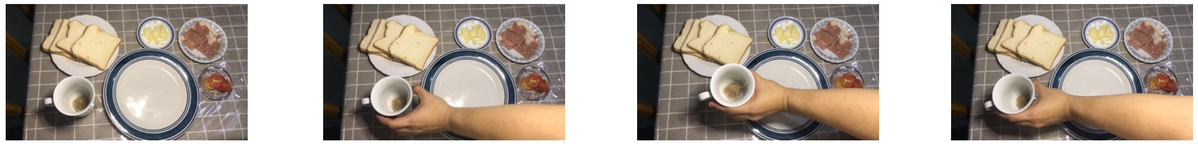
Event Missed!

Macro: Key events happen between sampled frames

Micro: Fine-grained actions below sampling rate

⚠️ 1-hour video at 1FPS: 3600 frames - only sample 64

What is wasted?



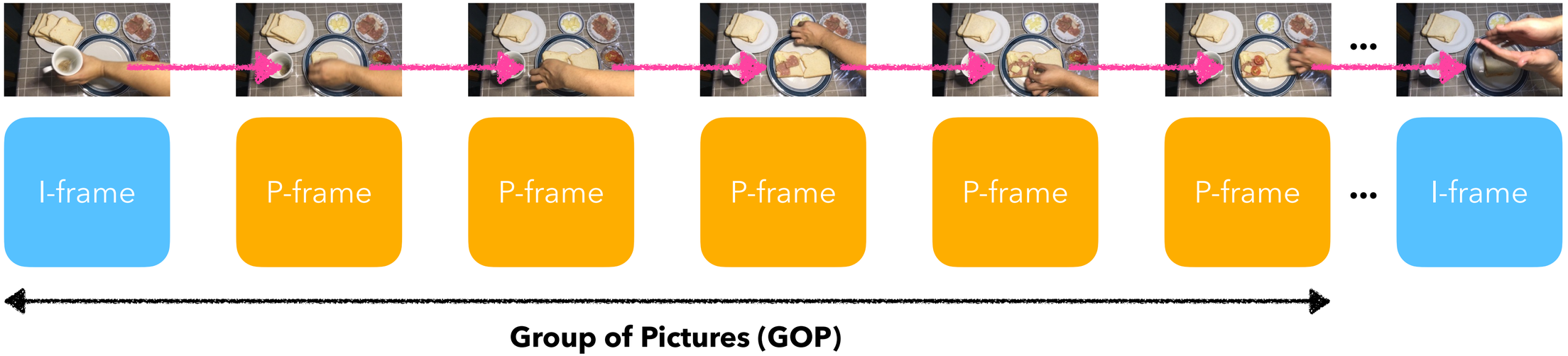
High redundancy between consecutive frames (even when downsampled to 1 FPS) → Only hand moving!

Using the same token budget for each keyframe: **suboptimal**

Key Insight: Information content scales with video duration, but the context window does not – smarter representation.

But Video Codecs Already Solved This!

Standard codecs only store what moves and what changes - not every single pixel



Intra-coded (I-frame): Full RGB image, encoded independently; serves as the **reference point** of the GOP.

Predictive (P-frame)*: Only stores changes from the previous frame: motion vectors + residuals.

block-wise optical flow

difference after motion compensation

* Bi-predictive (B-frames) ignored for explanation simplicity

* Video subsampled for easier visual understanding

Inside a P-frame: Motion Vectors + Residuals

Motion describes displacement, residuals captures what's left

How does the **recurrence relation** look like?

$$\hat{I}_i^{(t)} = \hat{I}_{i-\tau_i^{(t)}}^{(t-1)} + \delta_i^{(t)}$$

Reference frame at time $t-1$

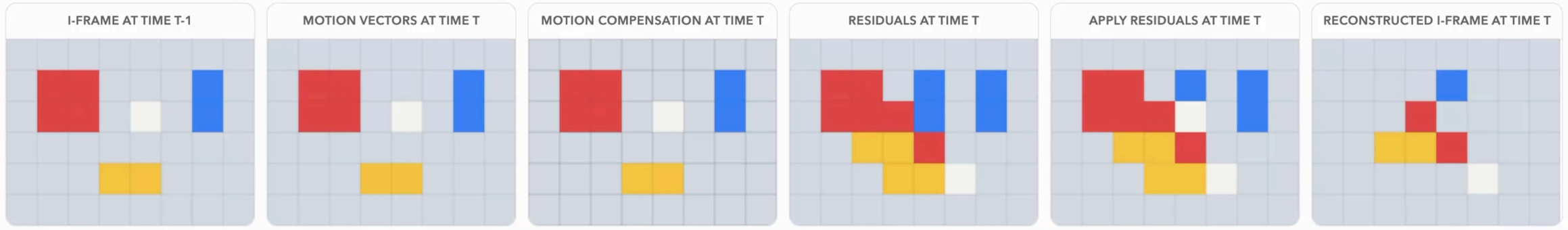
Residual at time t

Reconstructed frame at time t

Next **decoded RGB** frame

Motion vector from $t-1$ to t

Let us understand with an animation!

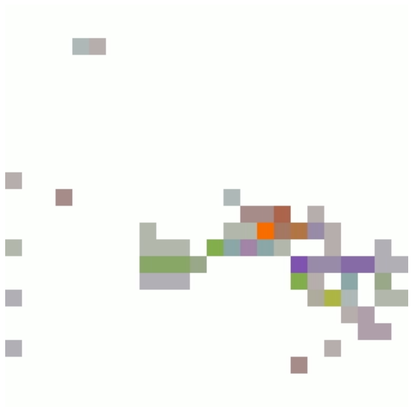


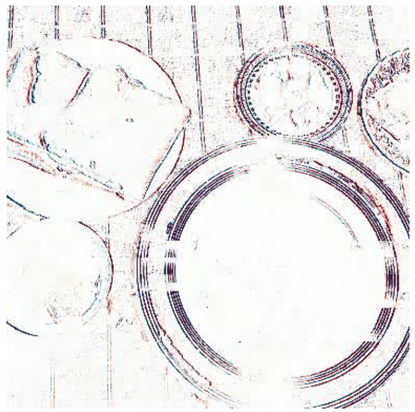
How do **motion vectors + residuals** look like in practice?

Inside a P-frame: Motion Vectors + Residuals

Motion describes displacement, residuals captures what's left

RGB Video | **Motion Vectors** | **Residuals**





Low motion (cooking)

High motion (sports)

Key Insight: Motion vectors and residuals are inherently sparser compared to full RGB regardless of scene complexity.

The Disconnect

Codecs compress intelligently – VideoLMs throw it all away!

What Codecs Give Us?

+ Structured sparsity (most P-frame data is near-zero)
+ Explicit motion and temporal change information
+ Compact representation (P-frames << I-frames in size)

What VideoLMs do?

- Fully decode everything back to RGB
- Encode every frame independently with same token budget
- Sample a fixed number of frames regardless of video length

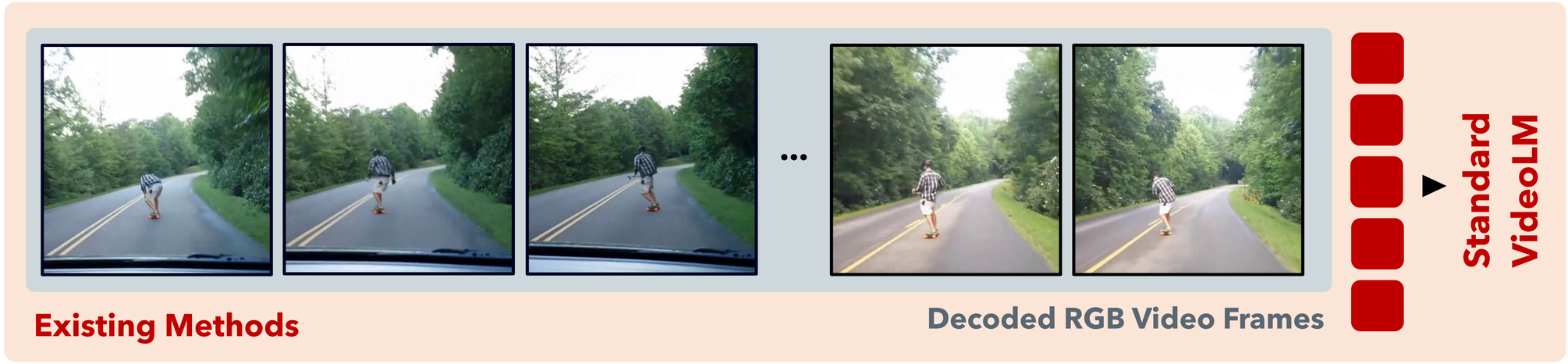
Example: Open-source VLMs have a 32K/64K token context window – fitting only **~64 RGB frames.** For a 1-hour video: **~0.12 FPS sampling**, while videos are captured at **20-30 FPS.**

Problem: Excessive computational & memory overhead for short videos + surpass the LLM context window for long videos.

Key Insight: What if we kept the natively sparse codec representation and fed it directly to the LLM?

Our Approach: CoPE-VideoLM

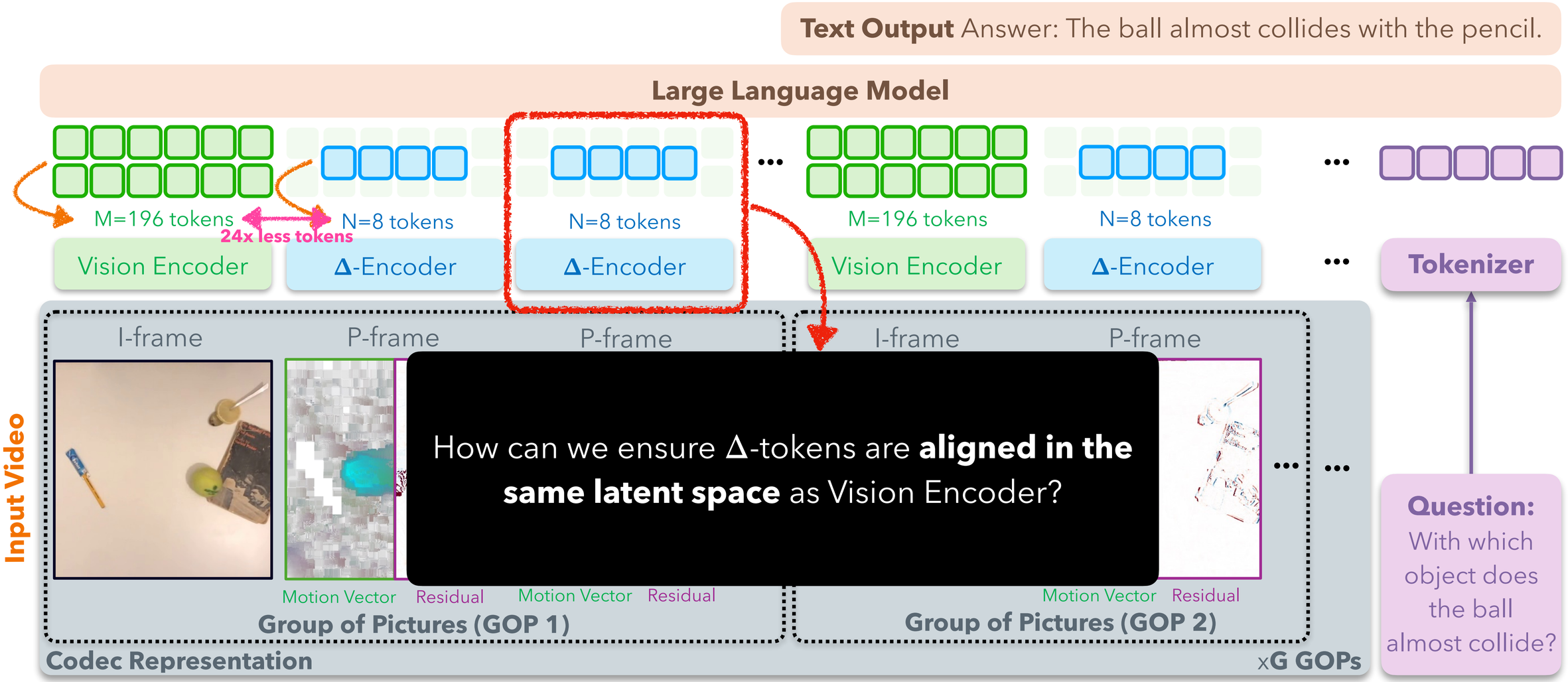
Enable denser temporal coverage at a fraction of the standard token count and runtime



Codec-aware tokenization framework, leveraging **inherent sparsity and structure** of video codecs.

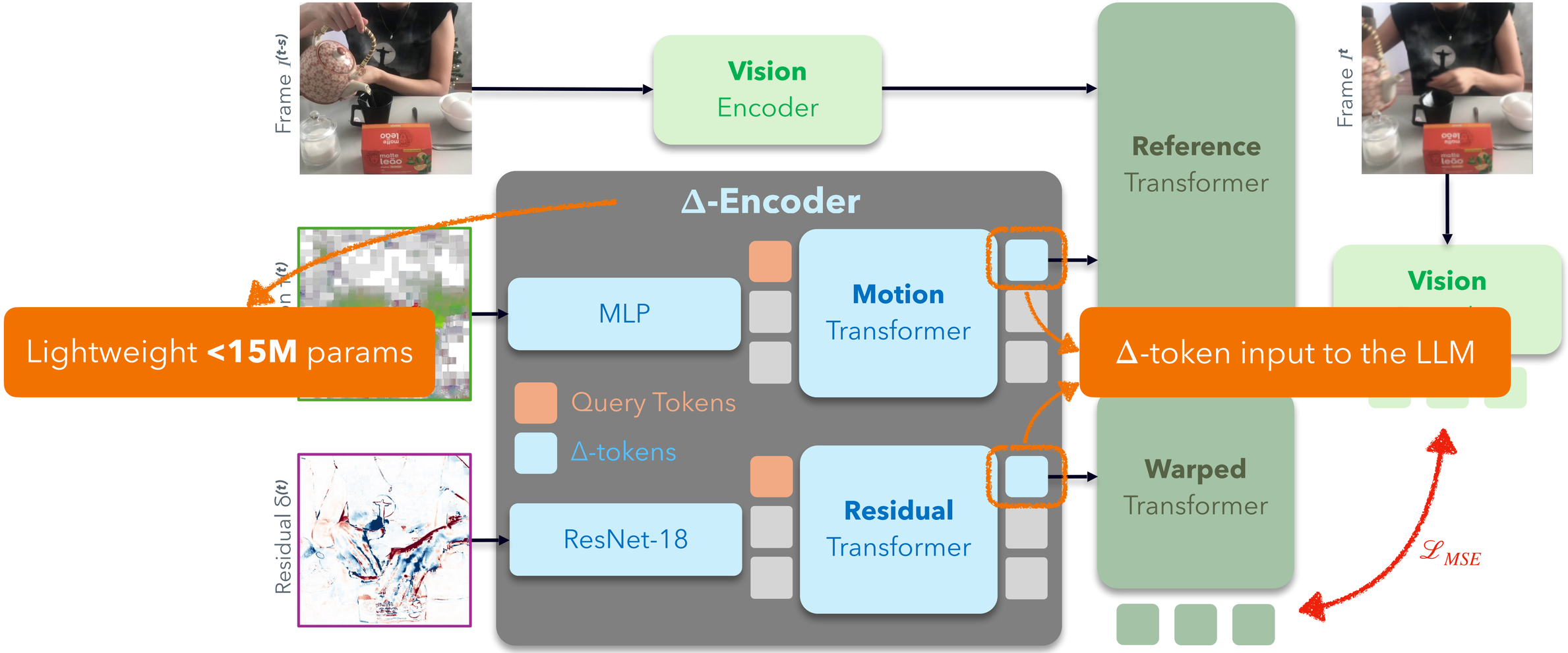
Overview of CoPE-VideoLM

Replace dense RGB image embedding with structured codec-based representation



How do we pre-train Δ-Encoder?

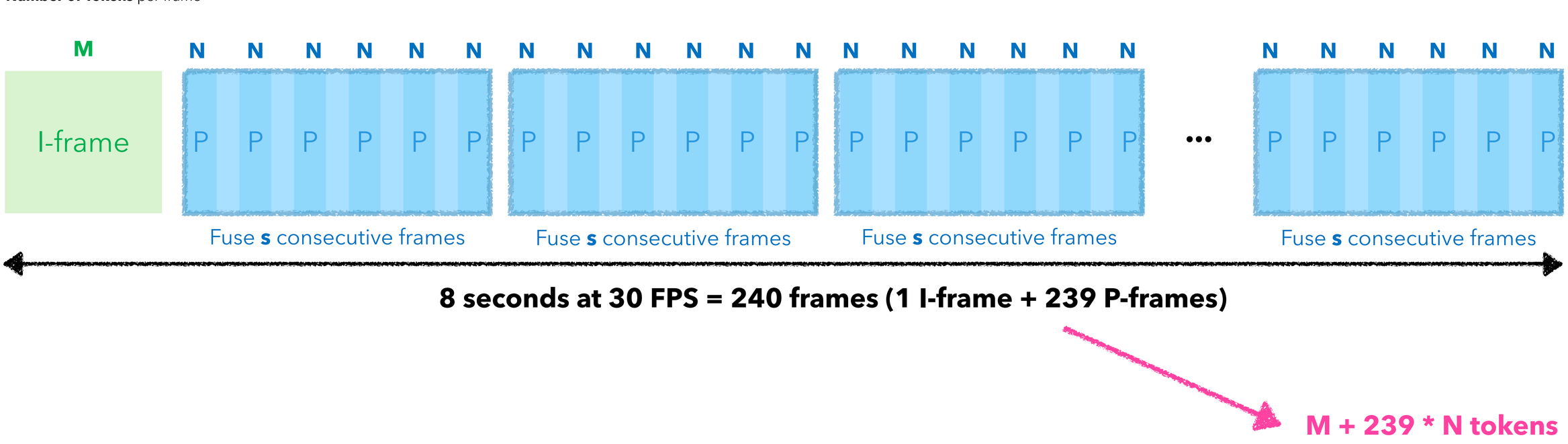
Align compact P-frame tokens with the vision encoder token space



Process P-frames through **two lightweight branches** designed to **extract and compress codec information**.

P-frame fusion

Do we really need to process every single P-frame independently?

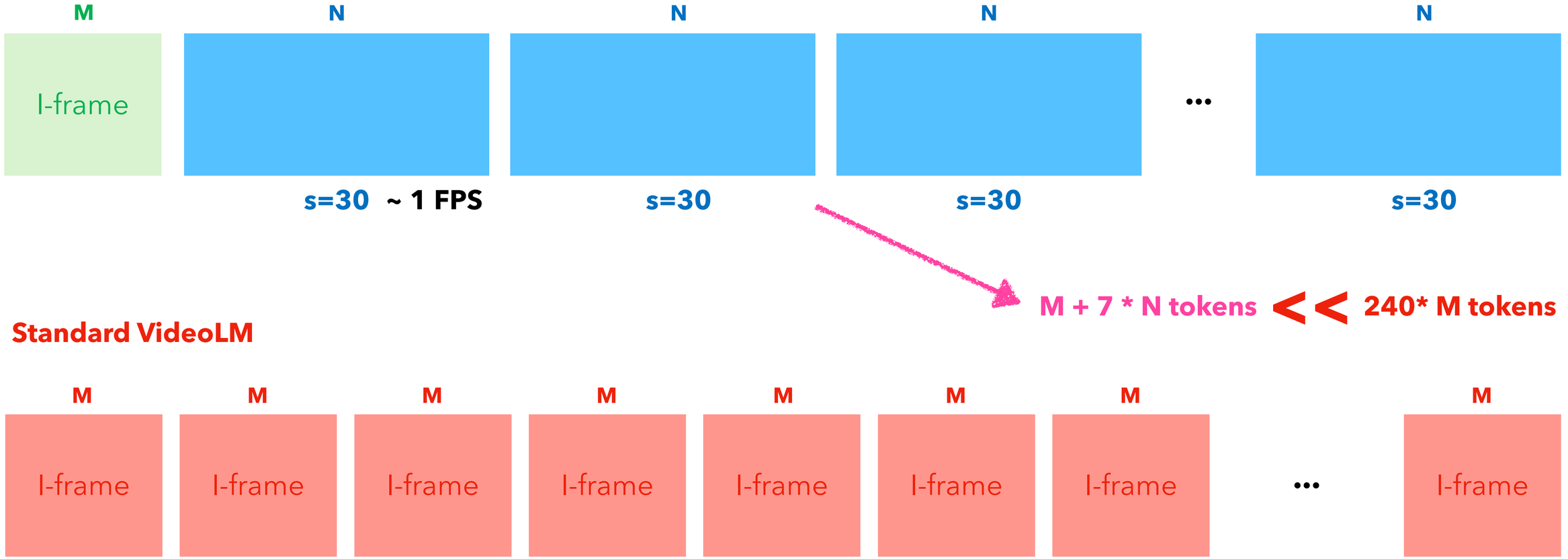


Sure, fine-grained action recognition may require this exhaustive temporal coverage

BUT, most video understanding tasks can be done with sparser coverage!

P-frame fusion

Do we really need to process every single P-frame independently?



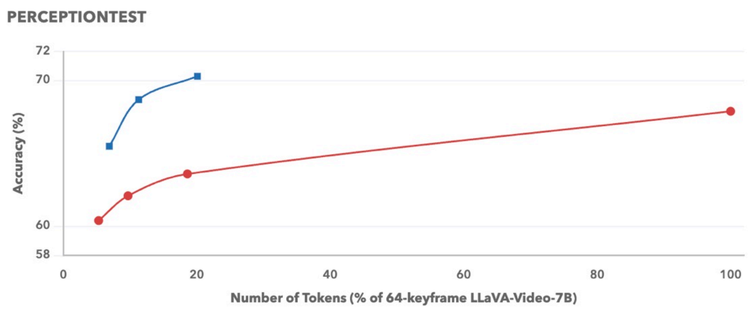
Trade **temporal resolution for efficiency** by **grouping** P-frames.

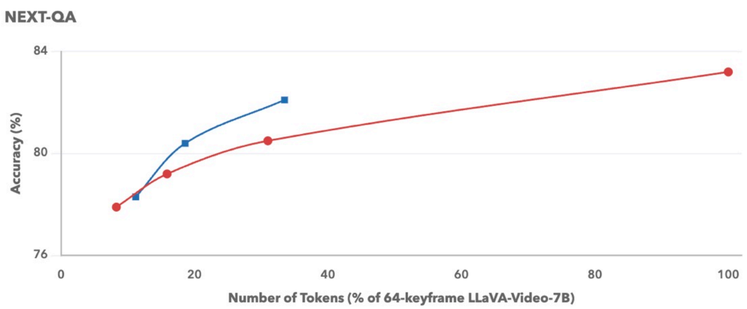
Does the Representation Hold Up?

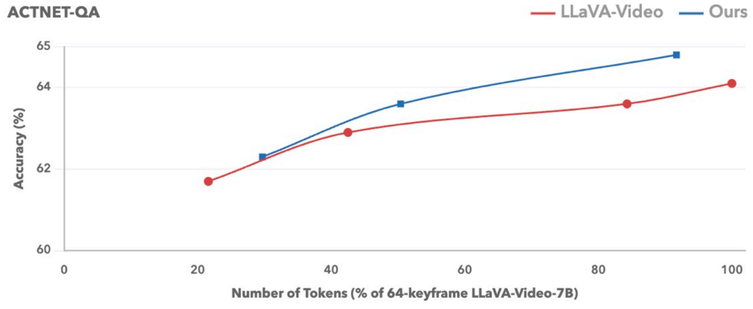
Time for Experimental Results!

Token Efficiency vs Video QA Accuracy

How much accuracy can Δ-tokens recover when added to sparse keyframe baselines?







At every keyframe density, adding Δ-tokens from P-frames **consistently improves accuracy** while barely increasing token count.

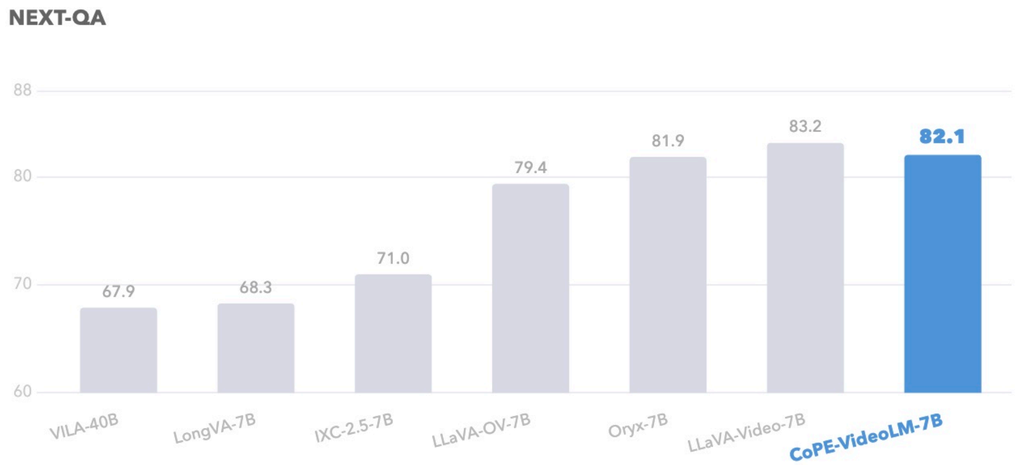
Even under aggressive compression, the model maintains strong performance with an order-of-magnitude token reduction.

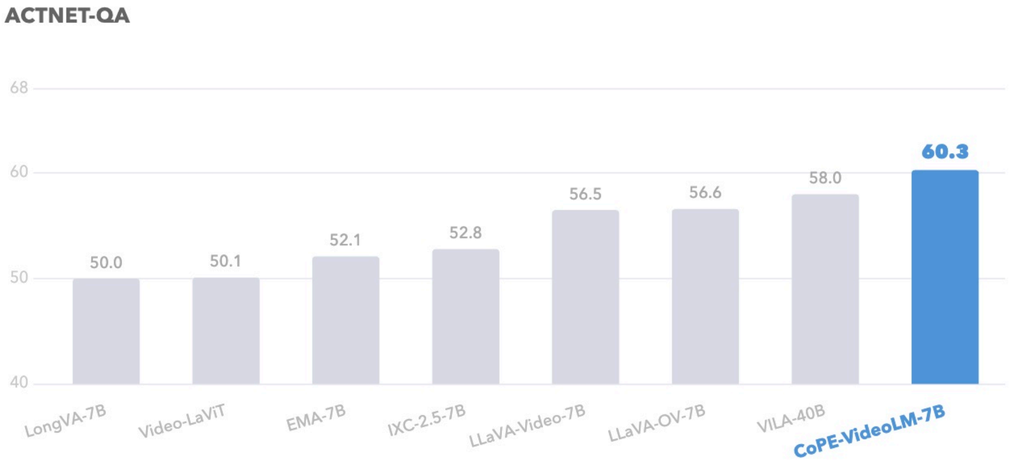
At higher frame densities (4KF per GOP), we not **only match but often surpass the performance of the 64-frame baseline**, despite using only a fraction of its tokens. → **Δ-encoder successfully captures motion and appearance cues critical for temporal reasoning.**

Our method preserves **fine-grained semantics** and advances the **Pareto-optimal frontier**.

Benchmark Evaluation

General Video Question Answering



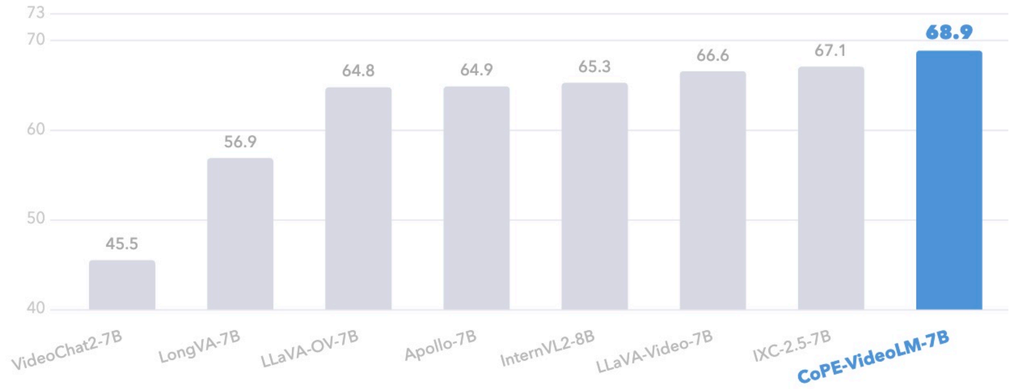


Codec primitives preserve the **semantic and temporal cues** required for diverse understanding tasks.

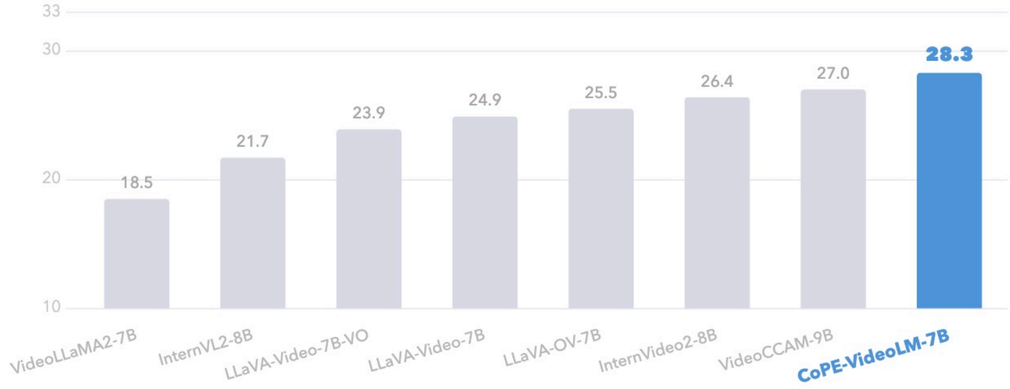
Benchmark Evaluation

Temporal Reasoning And Motion Understanding

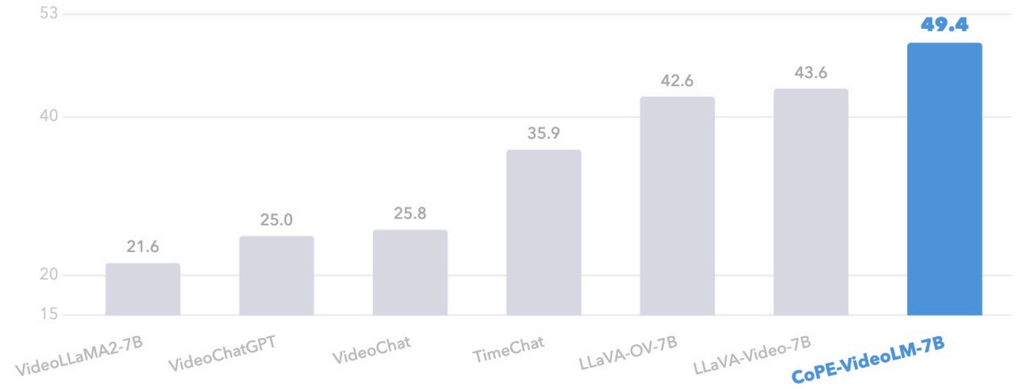
TEMPCOMPASS



TOMATO



CVRR-ES



MVBENCH

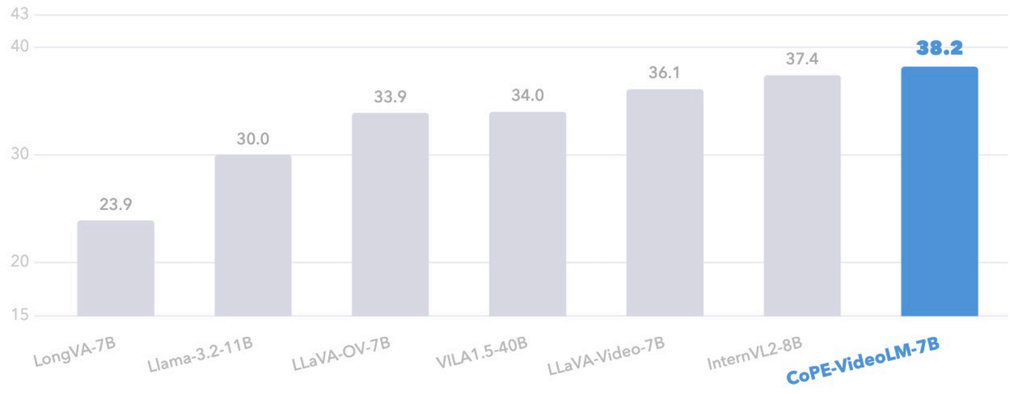
Explicit encoding of motion vectors and residuals: **stronger temporal signal** than dense RGB frame processing.

Benchmark Evaluation

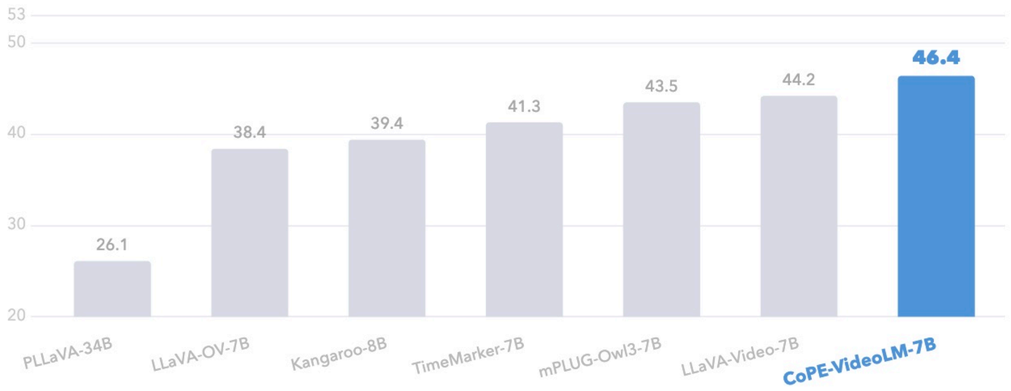
Long-form And Instruction Following

VIDEO-TT

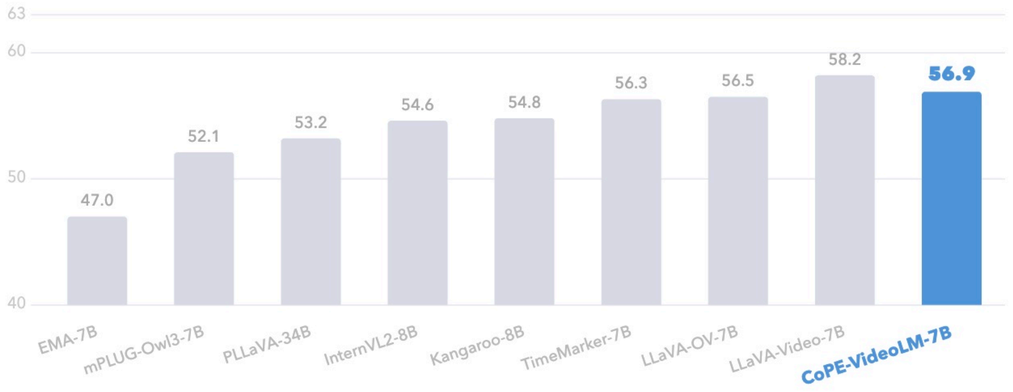
VIDEO-MMMU



LVBENCH



LONGVIDEOBENCH



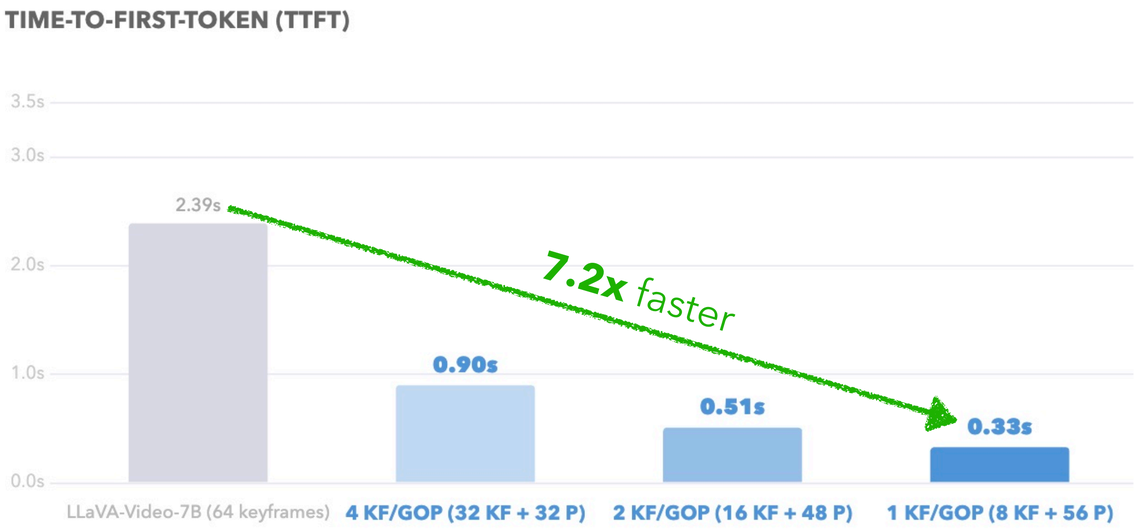
Compressing P-frames into compact Δ-tokens enables **more temporal coverage** within the **same token budget**.

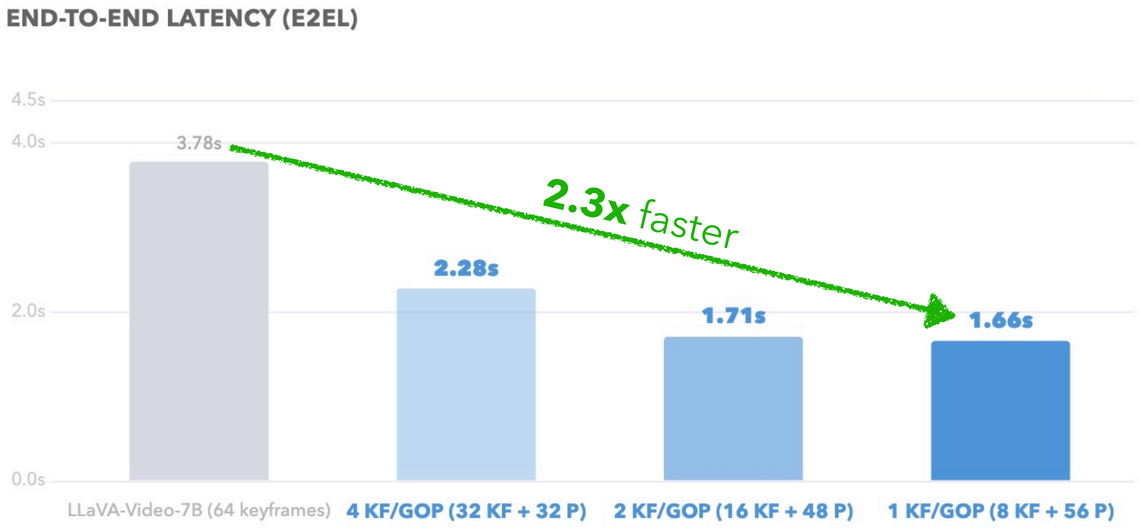
But Accuracy is Half The Story!

Runtime and Latency

What happens to inference speed when most frames skip the vision encoder?

We measure time-to-first-token and end-to-end latency for generating 64 text tokens on a single RTX 4090 at 1 FPS video input.



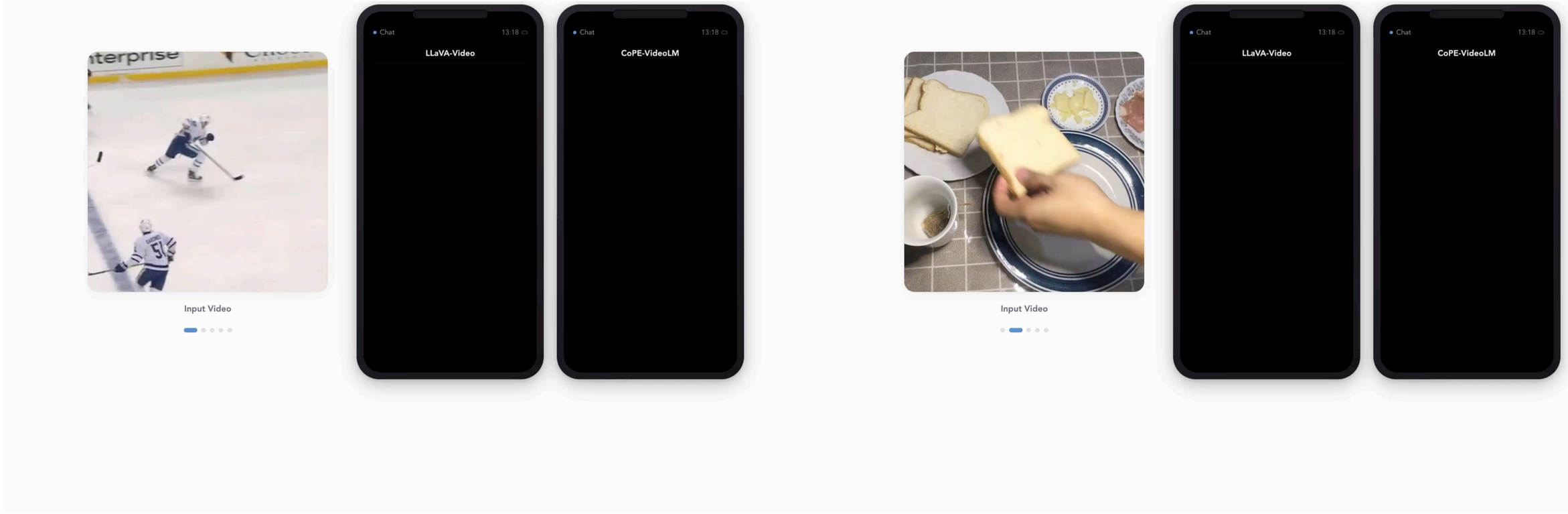


Since only I-frames require full vision encoding, both pre-fill time and sequence length shrink significantly.

Low TTFT is essential for **interactive user experience** and **real-time responsive** embodied agents.

Interactive Runtime Comparison

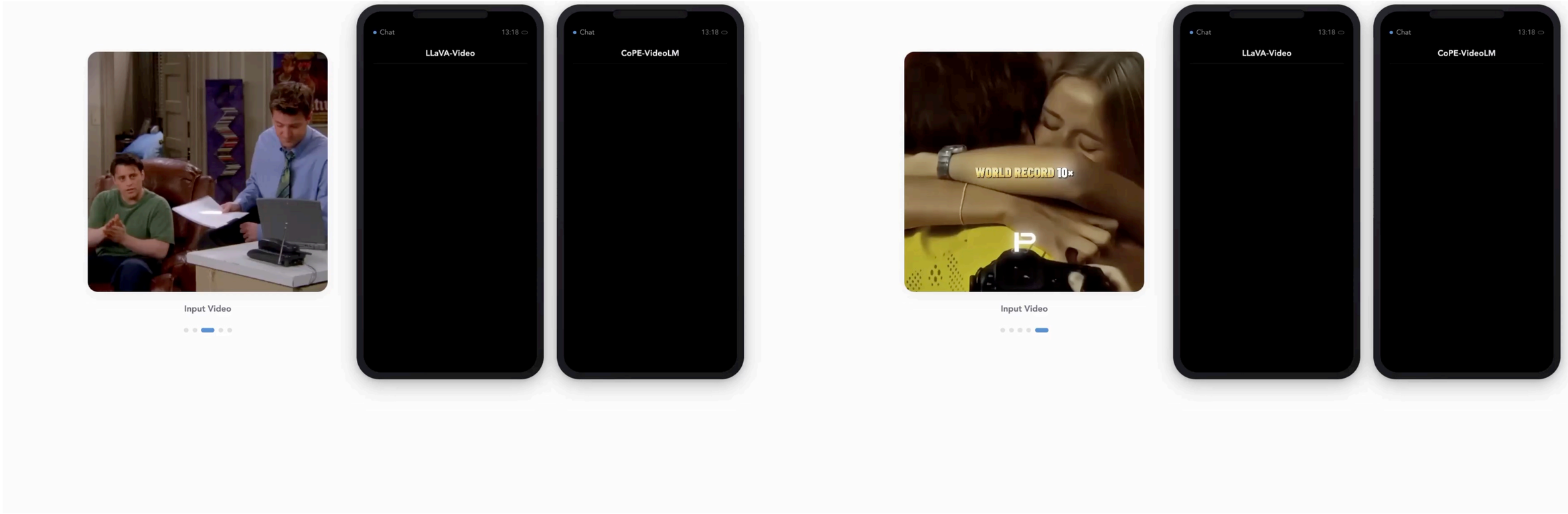
Same video, same question, same hardware: different input representation



CoPE produces accurate answers while achieving **consistent speedups** at **substantially reduced token** counts.

Interactive Runtime Comparison

Consistent speedup across diverse video content

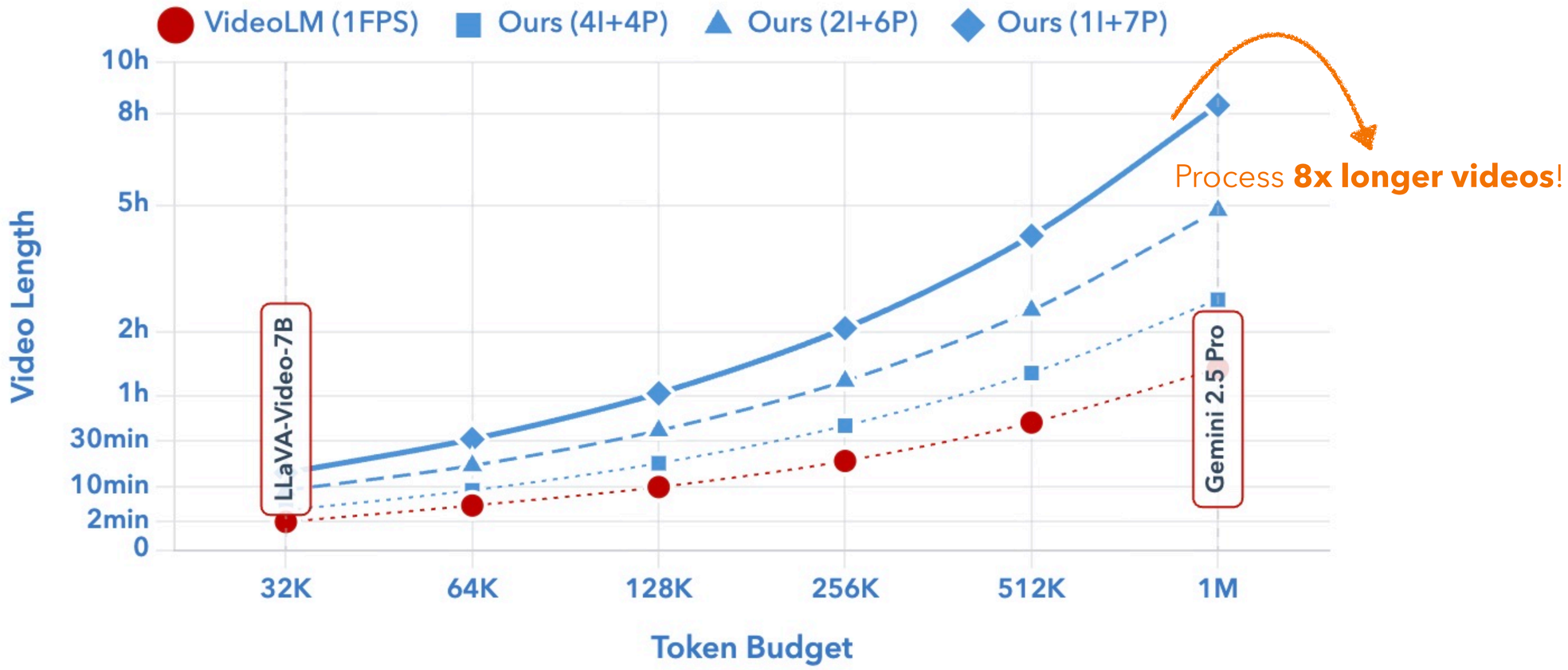


Codec-aware tokenization is not **just semantically sound** but is a necessity for **enabling fast inference**.

Scaling to Longer Videos

How far can compact Δ-tokens stretch the context window?

Standard dense RGB sampling saturates quickly, limiting coverage as memory constraints are rapidly encountered.



Compressing P-frames into compact Δ-tokens enables **more temporal coverage** within the **same token budget**.

Scale of Training Data

Structured compression yields stronger performance with smaller training corpora

Trained **only** on 1.39M videoQA samples – no image alignment data.

Against the most comparable configurations, our method **outperforms or remains on par across benchmarks** despite **using fewer samples.**

TRAINING DATA	TOTAL DATA	NEXT-QA	PERCEPTIONTEST	VIDEOMME
LLaVA-Video				
LLaVA-Hound	0.25M	64.4	51.4	54.1
+ LLaVA-Video-178K	1.58M	80.1	57.1	63.2
+ 3 QA datasets	1.64M	80.1	69.0	61.9
+ LLaVA-OV (Images)	2.74M	83.2	67.9	63.4
LLaVA-Video-178K (sampled)	1.08M	73.2	55.9	59.6
CoPE-VideoLM-7B (Ours)				
LLaVA-Video-178K + 3 QA datasets	**1.39M**	**82.1**	**70.3**	**61.9**

Same data, better results: gap is training data scale & composition, not codec-specific limitation.

Spatial Video Question Answering

Can codec primitives capture enough geometry for 3D reasoning?

We evaluate on SQA3D and ScanQA: both benchmarks require associating multi-view observations with 3D spatial structure.

METHOD	SQA3D (TEST)		SCANQA (VAL)				
	EM	EM-R	CIDER	BLEU-4	METEOR	ROUGE	EM
3D VLMs							
LLaVA-3D	55.6	57.6	91.7	14.5	20.7	50.1	27.0
Video-3D-LLM	58.6	-	102.1	16.4	20.0	49.3	30.1
Ross3D	63.0	65.7	107.0	17.9	20.9	50.7	30.8
2D VideoLMs							
InternVL2-8B	33.0	45.3	62.5	3.3	14.5	34.3	-
Qwen2-VL-7B	40.7	46.7	53.9	3.0	11.4	29.3	-
LLaVA-Video-7B (100% tokens)	48.5	-	88.7	3.1	17.7	44.6	-
CoPE-VideoLM-7B (Zero-shot, 25.78% tokens)	**46.5**	**49.8**	**70.9**	**7.1**	**14.7**	**38.2**	-
CoPE-VideoLM-7B (Fine-tuned, 25.78% tokens)	**56.6**	**59.3**	**96.9**	**14.9**	**19.1**	**46.4**	**27.5**

Without camera poses or point clouds, CoPE-VideoLM matches leading models **at 25% of the total token cost.**

Codec tokens carry **rich spatial signals** at a fraction of the usual budget.

Can we integrate motion + residual information in 3D VLMs to extract better spatial intelligence?

Comparison with Token Pruning

How do we perform against dense token compression methods?

Token pruning methods reduce the LLM's token count but still pay the **full image embedding cost** for every frame.

Our representation inherently encodes **only meaningful temporal changes** rather than removing information post hoc.

METHOD	ACTNET-QA	NEXT-QA	VIDEOMME
FastV	47.95	81.1	57.5
DyCoke	47.88	81.1	57.4
PLLaVA	47.59	81.0	56.9
VisionZip	45.42	78.5	54.2
LLaVA-Scissor	47.89	81.2	57.4
CoPE-VideoLM-7B (Ours)	**60.28**	**82.1**	**61.9**

All baselines use **50% token budget**; our method achieves up to **93% lower token usage** while being **7x faster in TTFT**.

Codec-native sparsity can provide a **strong starting point** for post-hoc pruning.

What makes CoPE-VideoLM work?

Dissecting A Few Key Design Choices

*Ablations trained on 60K samples instead of 1.39M samples, not directly comparable to main results.

Ablation: Two-Stage Training

What happens if we skip encoder pre-training and let LLM figure out from scratch?

Without pre-training, the model must simultaneously learn codec interpretation, feature alignment, and multimodal reasoning.

slowing convergence and **weakening** temporal understanding!

PRE-TRAIN Δ-ENCODER	FINE-TUNE LLM	PERCEPTIONTEST	NEXT-QA
1 Keyframe per GOP			
-	✓	63.45	74.56
✓	✓	**67.33**	**77.37**

Pre-training ensures that Δ-tokens inhabit a **well-structured embedding space**, while fine-tuning teaches the LLM how to **fuse the I- and P-frame tokens** together.

Spatially consistent pre-alignment bridges the codec and RGB domains.

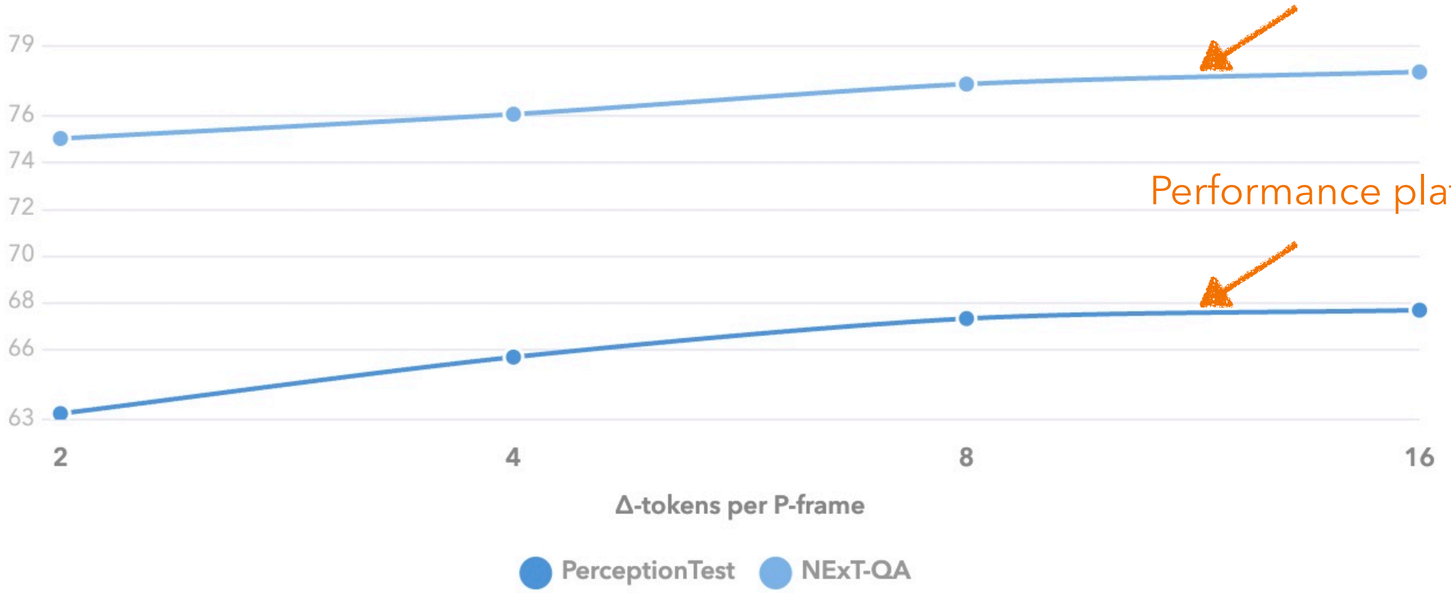
Can we train a unified vision encoder which can take both I- and P-frame as input?

Ablation: Varying the Number of Δ-Tokens

How many tokens does a P-frame need?

Fewer tokens: more aggressively compressed representation.

More tokens: allows the Δ-Encoder to retain finer motion and appearance cues from the codec primitives.



Performance plateaus **beyond N=8 tokens**

N=8 tokens suffice: codec primitives are **inherently compact**.

Could adapting the token budget per P-frame based on motion and task complexity push this further?

Ablation: Are Δ-tokens used by the VideoLM?

What if we keep the temporal structure but zero-out all codec information?

Run inference with 1 keyframe + 7 P-frames per GOP **but set all Δ-tokens to zero**

Preserve the native temporal order while stripping all motion and residual information

SAMPLING STRATEGY	PERCEPTIONTEST	NEXT-QA
1 Keyframe per GOP		
+ Δ-tokens = 0	64.41	74.21
+ Δ-tokens	**67.33**	**77.37**

LLM actively leverages Δ-tokens for **temporal reasoning** rather than **interpolating** between sparse I-frames.

Key Takeaways

Codec-aware tokenization: a practical and efficient foundation for future VideoLMs

Codec primitives are a natural fit for VideoLMs.

Motion vectors and residuals already encode temporal dynamics; there's no reason to discard this structure and re-process everything as dense RGB.

Massive efficiency gains with competitive accuracy.

Up to 93% fewer tokens, 86% faster TTFT and maintains performance across 14 diverse video understanding benchmarks.

The approach is general and scalable.

Standard transformer components, no architectural modifications to the LLM, and applicability beyond video QA to retrieval, robotics, and 3D understanding.

Decades-old video compression meets modern language models!

Key Takeaways

Codec-aware tokenization: a practical and efficient foundation for future VideoLMs

+ Codec primitives are a natural fit for VideoLMs.

+ Massive efficiency gains with competitive accuracy.

+ The approach is general and scalable.

Limitations & Future Work

? How can we add support for B-frames, thereby, modeling complex causal dependencies?

? Can we directly operate on sets of block-wise motion vectors + quantized DCT coefficients to stay closer to raw codec primitives?

? The fixed fusion window is suboptimal for videos with varying motion. Can motion + content adaptive fusion improve both efficiency and accuracy?

Thank You!

Questions?

CoPE-VideoLM: Leveraging Codec Primitives For Efficient Video Language Modeling

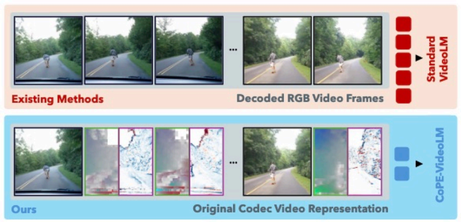
Sayan Deb Sarkar[1,2°], **Rémi Pautrat**[1], **Ondrej Miksik**[1], **Marc Pollefeys**[1,3], **Iro Armeni**[2], **Mahdi Rad**[1*], **Mihai Dusmanu**[1*]

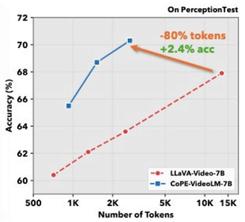
[1]Microsoft Spatial AI Lab, [2]Stanford University, [3]ETH Zurich

[°]Part of work done at Microsoft [*]Equal Supervision

Video Language Models (VideoLMs) enable AI systems to understand temporal dynamics in videos. To fit within the maximum context window constraint, current methods use keyframe sampling which often misses both macro-level events and micro-level details due to the sparse temporal coverage. Furthermore, processing full images and their tokens for each frame incurs substantial computational overhead. We address these limitations by leveraging video codec primitives (specifically motion vectors and residuals) which natively encode video redundancy and sparsity without requiring expensive full-image encoding for most frames. To this end, we introduce lightweight transformer-based encoders that aggregate codec primitives and align their representations with image encoder embeddings through a pre-training strategy that accelerates convergence during end-to-end fine-tuning. Our approach, *CoPE-VideoLM*, reduces the time-to-first-token by up to 86% and token usage by up to 93% compared to standard VideoLMs. Moreover, by varying the keyframe and codec primitive densities we maintain or exceed performance on 14 diverse video understanding benchmarks spanning general question answering, temporal and motion reasoning, long-form understanding, and spatial scene understanding.

Date: March 30, 2026
Correspondence: Sayan Deb Sarkar at sdsarkar@stanford.edu
Project: https://microsoft.github.io/CoPE





Project Page

Project Page: https://microsoft.github.io/CoPE/