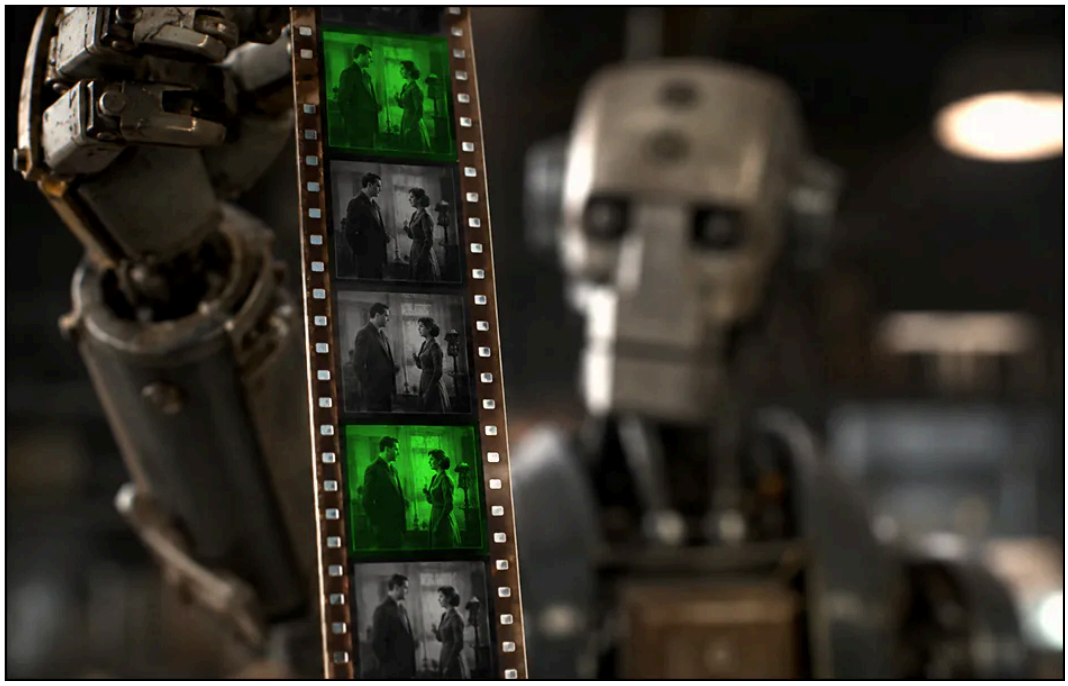
A Video Codec for AI-Generated Footage

Originally published at https://www.unite.ai/a-video-codec-for-ai-generated-footage/



As the all-you-can-eat era of AI draws to a close, an economical new approach to AI video generation promises notable savings in tokens and time.

The real cost of AI inference is bringing a new note of sobriety to the breakneck pace of the current AI revolution, with increased interest in rationalizing the cost of machine learning. Besides the potential of bringing AI in-house, and the general rise of private AI, VRAM-hungry and resource-gobbling machine learning routines will clearly need optimization too.

Video generation is perhaps the biggest offender, in this respect. If you've ever recompressed a movie or exported one out from a video-editing suite, you already know the toll that this particular (non-AI) task takes on your hardware – eating RAM and CPU cycles, and often blocking the machine for any other usage, unless measures are taken to limit the compression algorithm's impact on the host computer.

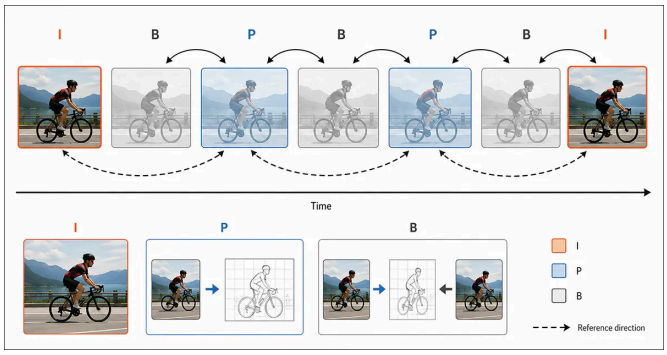
Therefore one need only imagine the extent to which the rise of AI video is replaying this 'power-mad' procedure in data centers around the globe. At that scale of operation, the smallest gains become immediately significant in the aggregate reckoning.

In the Frame

With this in mind, a new research offering from Shanghai, in collaboration with JD.com , is proposing a video codec aimed not at the rendering-out process (the process of compressing huge, raw frames into a smaller video file size), but at the actual AI video generation process itself.

A normal video codec works by not storing every frame as a full image, but by creating a smaller number of complete pictures, called *I-frames*, and then storing *changes between frames*.

For instance, if a person moves slightly in a video, the codec records only the parts of the frame that register that change, rather than rewriting the whole scene. These are *P-frames*, which are derived from earlier frames, and *B-frames*, which can also anticipate information in future frames:



Anatomy of a video codec: top row shows frames over time labeled I, P, and B, with I-frames fully stored in full color and P- and B-frames shown faded to indicate reconstruction; arrows indicate whether frames use earlier frames, later frames, or both; lower panels depict a fully stored frame (I-frame, left), a frame built from a previous frame (P-frame, second-left), and a frame built from both earlier and later frames (B-frame, right).

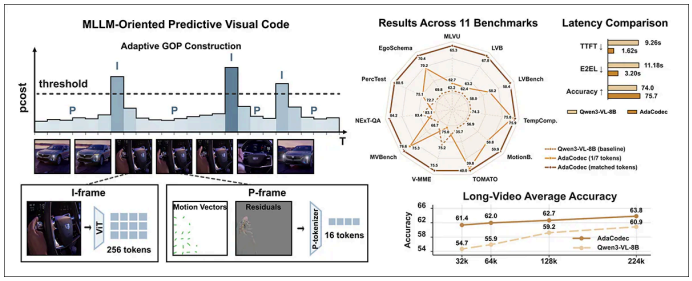
This reuse of nearby information is why video files stay small, with most frames operating not as new images, but as instructions describing how the previous frame has changed. Therefore Iframes constitute 'full-fat', space-hogging uncompressed (or minimally compressed) images, with the frames between them constituting only the difference between Iframes (and between themselves).

When *every single frame* is a full uncompressed image, the movie essentially has *no compression*. Saving a movie in this way, as uncompressed video, would result in a 2-hour movie needing near (or more than) a terabyte of disk space. Yet this is how AI makes movies† – by dedicating equal resources and tokens to each and every frame, when it's calculating how to formulate the video.

Economy of Scale

The new work, titled *AdaCodec: A Predictive Visual Code for Video MLLMs*, instead expends full visual tokens *exclusively on reference frames* (Iframes), with all interstitial frames rendered as 'compact P-tokens' – a paradigm clearly taken from the traditional compression employed by historical 'real world' video codecs.

After this internal compression has taken place, the genAI video can then be compressed normally, and, in theory, all the savings are server-side:



Overview of AdaCodec. Left, videos are divided into adaptive groups of pictures, with full I-frames reserved for moments that are difficult to predict and intermediat P-frames represented using compact motion and residual information. Right, the resulting system matches or exceeds Qwen3-VL-8B across eleven benchmarks, maintains higher long-video accuracy across token budgets, and reduces response latency while processing substantially fewer video tokens. Source

The savings, according to reported results from tests for AdaCodec, are worth pursuing; the paper states that the system outperformed the unmodified Qwen3-VL-8B model across every major benchmark, while using the same amount of processing; and still matched or exceeded that model's performance after cutting video tokens by an impressive approximate 86%.

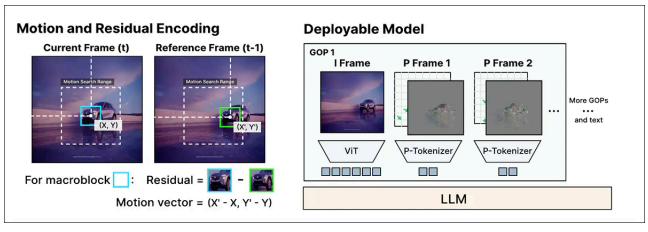
The authors state*:

> *'We draw inspiration from predictive coding, where a system transmits errors from a prediction rather than the raw signal. This principle has biological grounding: the visual system is thought to encode prediction errors, the mismatch between expected and observed input, rather than the input itself.*
>
> *'Modern video codecs use the same residual-coding idea in engineering: reference frames carry full content, while predictive frames carry motion and residual signals relative to a reference.*
>
> *'These systems have different objectives, but they share the same conditional structure: when nearby samples are redundant, the channel should carry what prediction fails to explain.*
>
> *'Standard codecs, however, optimize for bitstreams and human-viewable reconstruction, not for visual tokens consumed by an LLM. We therefore redesign this mechanism as an MLLM interface for video understanding.'*

The new work, written by 11 researchers from Shanghai Jiao Tong University, Shanghai Innovation Institute, and JD.com, comes with an associated project page, with release of source code promised.

Method

As discussed, instead of treating every frame as a completely new image, the system looks for what changed between one frame and the next. On the left, in the image below, we see a small region of the current frame that's matched against the most similar region in an earlier frame:



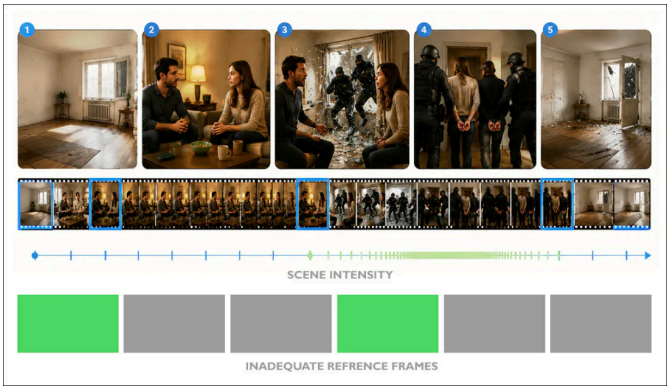
Schema overview for AdaCodec.

The distance between the two locations becomes a motion vector, while any remaining visual differences become a *residual*, with these compact descriptions replacing the need to store a full image.

On the right, we see the resulting information fed into the AI model: important reference frames are still processed as complete images, but the intervening frames are represented by much smaller *motion-and-residual* tokens – apparently allowing the model to retain enough information to parse the video, while processing notably less visual data.

One interesting challenge is deciding which frames deserve to be stored in full: traditional video codecs usually place reference frames at regular intervals, whether they are needed or not. AdaCodec, instead, tries to identify the moments that matter most.

For instance, consider a scene mostly depicting a static conversation between two people in an apartment – and suddenly a SWAT team bursts in through the window. Immediately the camera views and number of editing cuts will ramp up and require much more data than a regularized reference frame interval will provide:



This is the logic behind variable compression (variable bitrate) compression methods, which analyze source video for such 'busy' periods, and assign greater data where it is needed – at no small cost of time and resources.

In AdaCodec, if a frame can be predicted accurately from nearby frames, the system continues using compact motion-and-residual tokens; if the scene changes substantially (i.e., the SWAT example above, or something less dramatic), a full reference frame is inserted. This allows more of the available processing budget to be spent on important visual information instead of being distributed evenly across the entire video.

Data and Tests

In testing, the researchers used the aforementioned Qwen3-VL-8B as the base model and evaluated AdaCodec across eleven benchmarks spanning three areas of video understanding: *long-video performance* was assessed with MLVU, LongVideoBench, and LVBench; *temporal understanding* with TempCompass, MotionBench, and TOMATO; and *general video understanding* with Video-MME, MVBench, NExT-QA, PerceptionTest, and EgoSchema.

Open source models tested were InternVL3.5-8B; Keye-VL-1.5-8B; GLM-4.1V-9B; MiniCPM-V-4.5-8B; Eagle2.5-8B; PLM-8B; LLaVA-Video-7B; VideoChat-Flash-7B; Molmo2-8B; and Molmo2-O-7B.

The GPT-5, Gemini, and Claude variations appear in the table below only as comparison baselines. CoPE-VideoLM-7B and ReMoRa-7B are earlier video-language models that reduce visual-token usage through codec-inspired compression, making them the closest direct competitors to AdaCodec:

Method	Long-video			Temporal			General				
	MLVU	LVB	LVBench	TempComp.	MotionB.	TOMATO	V-MME	MVBench	NExT-QA	PercTest	EgoSchema
Closed-source models											
GPT-5 [58]	77.7	72.6	65.2	80.4	65.4	53.0	86.9	74.1	86.3	79.4	75.6
GPT-5 mini [58]	69.1	69.7	54.7	74.9	59.9	44.1	82.3	66.5	83.2	72.0	70.9
Gemini 3 Pro [59]	75.7	75.9	77.0	82.8	62.6	48.3	87.5	70.4	84.3	77.6	68.9
Gemini 2.5 Pro [60]	81.5	76.8	75.7	81.9	62.0	48.6	87.8	70.6	85.3	78.4	72.2
Gemini 2.5 Flash [60]	75.1	73.1	64.9	80.2	59.3	39.1	84.2	67.0	81.8	74.7	70.2
Claude Sonnet 4.5 [61]	64.0	65.1	50.5	72.8	58.5	39.6	80.5	62.1	79.2	64.3	73.1
Open-source models											
InternVL3.5-8B [62]	53.2	62.1	43.4	70.3	56.6	24.6	68.6	72.1	81.7	72.7	58.6
Keye-VL-1.5-8B [26]	53.8	66.0	42.8	75.5	55.1	33.0	**76.2**	56.9	75.8	64.2	56.3
GLM-4.1V-9B [63]	56.6	65.7	44.0	72.3	59.0	30.0	75.6	68.4	81.3	74.2	62.6
MiniCPM-V-4.5-8B [64]	60.6	63.9	50.4	72.7	59.7	29.8	73.5	60.5	78.8	70.9	49.6
Eagle2.5-8B [65]	60.4	66.4	50.9	74.4	55.7	31.0	75.7	74.8	85.0	81.0	**72.2**
PLM-8B [66]	52.6	56.9	44.5	72.7	61.4	33.2	65.4	**77.1**	84.1	**82.7**	68.8
LLaVA-Video-7B [67]	52.8	58.2	44.2	66.6	54.2	24.9	69.7	58.6	83.2	68.8	57.3
VideoChat-Flash-7B [68]	56.0	64.7	48.2	69.4	60.6	32.5	69.7	74.0	85.5	76.5	51.3
Molmo2-8B [57]	60.2	67.5	52.8	73.4	**62.2**	39.6	75.8	75.9	**86.2**	82.1	62.0
Molmo2-O-7B [57]	55.2	63.7	49.6	73.0	60.6	36.2	69.2	74.8	84.3	79.6	56.8
Codec-aware video MLLMs											
CoPE-VideoLM-7B [30]	-	56.9	46.4	68.9	-	28.3	69.4	61.9	82.1	70.3	-
ReMoRa-7B [31]	-	60.8	-	-	-	-	64.4	-	84.2	67.7	-
Ours (AdaCodec on Qwen3-VL-8B)											
Qwen3-VL-8B [46]	62.2	62.4	58.0	74.3	56.9	35.7	75.2	68.7	83.4	72.7	69.8
AdaCodec (1/7 token budget)	62.7 +0.5	63.2 +0.8	58.2 +0.2	75.8 +1.5	58.8 +1.9	39.8 +4.1	75.0 -0.2	75.3 +6.6	83.1 -0.3	75.1 +2.4	70.2 +0.4
AdaCodec (comparable token budget)	**65.3** +3.1	**67.8** +5.4	**58.4** +0.4	**75.9** +1.6	59.9 +3.0	**40.0** +4.3	75.5 +0.3	76.6 +7.9	84.2 +0.8	80.5 +7.8	70.4 +0.6

Main results across eleven benchmarks covering long-video understanding, temporal reasoning, and general video understanding. Higher scores indicate better performance. LVB = LongVideoBench; V-MME = Video-MME. Bold and underlined values indicate the highest and second-highest scores among open-source models. Scores for closed-source models were taken from official reports where available, with missing results sourced from Molmo2, or evaluated by the authors.

To ensure a fair comparison, the same number of visual tokens was allocated to both AdaCodec and the standard Qwen3-VL-8B system, allowing the results to reflect the effectiveness of the compression approach, rather than any differences in computational resources.

At the most aggressive setting, AdaCodec reduced visual-token usage by about 86% while still matching or slightly exceeding the baseline system on long-video, temporal, and general video-understanding tasks.

When the saved tokens were reinvested into processing more video frames, performance improved across every long-video benchmark and every temporal benchmark, with gains reaching +5.4 points on LongVideoBench and +4.3 points on TOMATO, while also producing some of the strongest open-source results in the study.

Conclusion

Though projects of this nature are usually aimed at hyperscale providers, this is the kind of effort that will be of interest to hobbyists and SMEs alike, as part of a potential new 'public asceticism' around local, rationalized AI deployment.

In communities such as r/stablediffusion, this is very old news, as every major open source release that arrives there is regularly tormented into a hyper-optimized (GGUF, quantized weights, etc.) version capable of running, with some patience, on lower-end graphics cards.

If the ‘performative stage’ of this third AI upsurge is indeed over, and on the assumption that corporations will be repelled by the true costs of inference, then initiatives such as AdaCodec may comprise part of a coming ‘grand optimization’.

† *This is not the same as rendering out the video to a user-friendly format/file-size; rather, it deals with the internal generation and collation of frames that takes place inside the AI model at inference time.*

* *My conversion, within reason, of the authors’ inline citations to hyperlinks.*

First published Thursday, June 4, 2026

Writer on machine learning, domain specialist in human image synthesis. Former head of research content at Metaphysic.ai, until its dissolution into DNEG's Brahma.ai.
Portfolio site: martinanderson.ai
Contact: martin@martinanderson.ai

Discover More