TENSOR CLOCK: PUTTING ELAPSED TIME INSIDE VIDEO LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision language models are now widely applied to video understanding. Questions about how fast an object moves or when an event happens, however, expose a gap: when frames are sampled non-uniformly, as a fixed frame budget requires, frame order no longer determines elapsed time, so identical frames can imply different speeds and the same motion can occur at different moments. Timestamps written into the prompt do not close this gap; a model trained with true frame-pair timestamps in its prompt still cannot tell apart rates that differ only in timing.

This paper introduces *Tensor Clock*, a mechanism that lets elapsed time control the internal computation of a video language model rather than only appear in its input. Built on Qwen3.6-35B-A3B, it derives from the source timestamps of each frame pair the within-pair interval, the observation time, and the time elapsed since the previous block. These quantities set seconds-based rotary positions, drive elapsed-time decay in a measured subset of Gated DeltaNet recurrent heads, and condition the value features. Counterfactual training data, in which pixels are fixed and only the timing changes, force the answer to depend on time.

Against a control trained with identical data, text timestamps, and schedule, Tensor Clock raises counterfactual rate accuracy from 0% to 95.8% and improves event-localization F1 by 0.07 and 0.03 on two event-timing tasks. Intervening on a trained model shows how the two carriers divide the work: rate comparison depends entirely on Tensor Clock, whereas event timing is carried by the text timestamps and further sharpened by Tensor Clock.

1 INTRODUCTION

Vision language models answer questions about what happens in a video by sampling a fixed number of frames, usually at a uniform stride, and writing each frame's timestamp into the prompt. For questions about content and order this suffices, because only which frame comes before which matters. Questions about how fast something moves or when an event happens depend instead on how much time separates the frames, and a uniform stride hides that dependence because every interval is the same. The stride cannot stay uniform in practice: a fixed frame budget is spent well only by sampling densely where motion is fast and sparsely where little changes, so the interval between frames varies across and within videos, and frame order no longer determines elapsed time. As figure 1 shows, the same frames in the same order then describe different motion, depending only on the timing attached to them. Answering such questions requires physical time, by which we mean source-video time with a known correspondence to capture time.

Existing models make timing visible without making the computation depend on it. One line writes time into the input, as absolute-time tokens (Guo et al., 2024), textual timestamps (Bai et al., 2025a), or interleaved timestamp tokens with a relevance-aware sampler (Yuan et al., 2025); under uniform sampling the timestamps and the frame indices carry the same information up to a constant per video, so a model trained on such data can answer without reading them. A second line moves time into the tensors: Qwen2.5-VL aligns its temporal rotary coordinates to seconds (Bai et al., 2025b), which changes where a frame sits in position space but leaves the recurrent state and the value pathway unaware of elapsed time, and is likewise trained on uniformly sampled data. In neither line is the model ever placed where the same pixels demand different answers because the timing differs.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

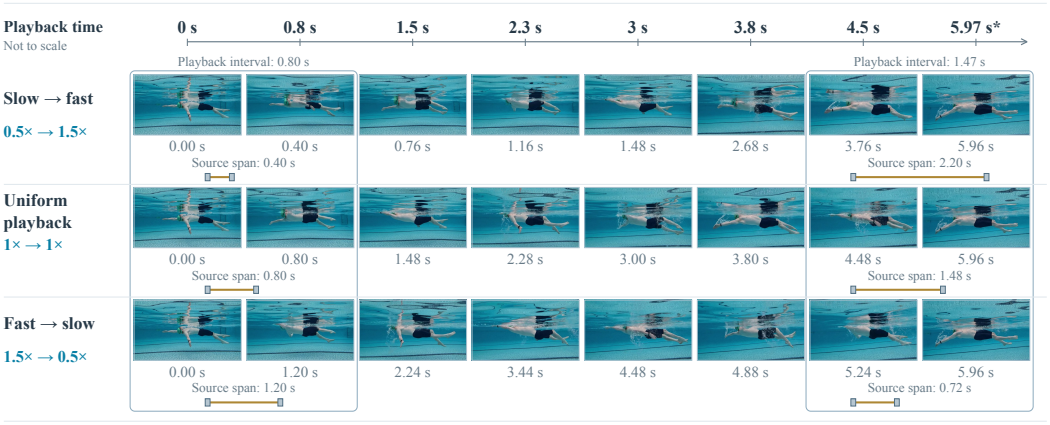


Figure 1: Same action, different timing. One swimming clip is played back three ways that share the same first and last source frames and are viewed at the same playback times; only the source span between neighbouring frames differs (0.40, 0.80, and 1.20 s for the first pair). Frame order is identical in all three, yet at a matched viewing time the action has progressed by different amounts, so the timing, not the order, carries the speed.

This paper introduces *Tensor Clock*, a mechanism that lets elapsed time control the internal computation of a video language model. A modern backbone already has three places where time can enter: Qwen3.6-35B-A3B (Qwen Team, 2026) interleaves attention, which reads positions, with Gated DeltaNet (GDN) recurrence (Yang et al., 2025), whose state decays through a gate, and both write through value projections. Tensor Clock derives the observation time and the time elapsed since the previous observation from the source timestamps of each frame pair and routes them to these three places (figure 2): observation times set seconds-based rotary positions, elapsed time drives the decay gate of the GDN heads whose measured memory spans more than one observation, and both condition additive value features. How much a selected head forgets between two observations is then set by the seconds that passed, not by the number of tokens in between. The model is trained on counterfactual data in which the pixels are fixed while the timing changes, and compared with a control trained identically but with native positions, recurrence, and values.

Against that control, Tensor Clock raises counterfactual rate accuracy from 0% to 96% and improves event-timing F1 by 0.07 and 0.03 on two tasks, while general video ability stays within one point. Intervening on the trained model shows how the two carriers divide the work: rate reasoning follows the timing in the tensors and event timing follows the timestamps in the text. Under uniform sampling the two models agree; the advantage of Tensor Clock appears when observations are irregular, which is where a fixed frame budget places them. In summary:

- **Tensor Clock**, which routes observation time and elapsed time from source timestamps to the positions, recurrent decay, and values of a hybrid attention–recurrence backbone, so that the computation itself depends on physical time.
- **Measured selection of the recurrent pathway**: a gate-based lookback measurement of the pretrained model picks the GDN heads whose memory spans more than one observation, and only these receive elapsed-time decay.
- We show that rate reasoning under irregular observation becomes possible only when elapsed time enters the computation and not the prompt alone: a control given identical timestamps in text never departs from zero. Intervening on the trained model, we further show that the two carriers serve different operations, rate reasoning reading time from the tensors and event timing from the text.

2 Frames Are Not Seconds

Observations and their times. A video is decoded into n sampled frames with source timestamps $\tau_1 < \cdots < \tau_n$, in seconds from the start of the video. Qwen3.6-35B-A3B, the backbone used

throughout, folds consecutive frames into pairs before its language model sees them (section 3.1), so the language model receives $K = n/2$ observation blocks, indexed $k = 0, \ldots, K-1$; block k holds the two frames with source times $t_{k,a} = \tau_{2k+1}$ and $t_{k,b} = \tau_{2k+2}$. From these we define

$$\Delta t_k^{\text{pair}} = t_{k,b} - t_{k,a}, \qquad t_k = t_{k,b}, \qquad \Delta t_k^{\text{state}} = t_{k,b} - t_{k-1,b} \quad (k \geq 1). \tag{1}$$

The first is the interval between the two frames of one observation; any speed read from the displacement inside a block is that displacement divided by this interval. The second is the time at which the observation is complete, which is where the observation sits on the timeline of the video. The third is the time elapsed between one observation and the next, which is how much a memory of the previous observation must age; the first block has no predecessor and receives a fixed reference value (section 3.2). We call the full schedule $\{(t_{k,a}, t_{k,b})\}_k$ the *timing* of a video, and each route by which the timing reaches the model a *carrier*: the timestamp text in the prompt is one carrier, and the position, decay, and value operations of section 3 are the others. *Uniform sampling* decodes frames at a constant stride of g seconds. Under it the two intervals are constant, $\Delta t_k^{\text{pair}} = g$ and $\Delta t_k^{\text{state}} = 2g$, and the observation time is affine in the block index, $t_k = t_0 + 2kg$.

Why uniform sampling hides timing. Uniform sampling matters because it is how nearly all training and evaluation data are produced, and under it timing is redundant with the block index: with $t_k = t_0 + 2kg$, a rotary phase θt_k is the index phase θk with its frequency rescaled by the per-video constant $2g$, and a decay driven by $\Delta t_k^{\text{state}} \equiv 2g$ depends only on the number of blocks up to one gain per video. A model that knows the span and the sampling rule can thus recover time from the index; nothing in uniformly sampled data rewards reading the timestamps, and evaluation on it cannot tell two ways of supplying time apart. Natural-video benchmarks are in this regime: in the four we audited the source frame rate is constant (appendix C), so a fixed index stride is exactly this schedule. A fixed frame budget, however, cannot keep the stride uniform: the frozen vision encoder reads within-block displacement only in a narrow band that most natural-video benchmarks fall below under 64-frame uniform sampling (appendix C), so the interval has to be chosen per video or per segment. Then Δt_k^{pair} and $\Delta t_k^{\text{state}}$ vary within a video, the index carries no fixed relation to time, and clips with identical decoded frames may differ in every quantity of equation (1). This paper therefore assumes a fixed observation budget under which the sampler chooses which source frames to decode, and asks what the model needs after that choice.

Two tests of time use. Section 4 has to decide whether a model uses physical time, so we fix the criteria here. Two complementary tests bear on the question. The first holds the decoded frames fixed and changes their timing; a model that uses time should change its timing-dependent answers, in the direction implied by the new timing. We call this *retiming response*; it is the test that separates a model that reads time from one that reads the index. The second changes which source frames are decoded while the underlying process and its labels stay the same; a model that uses time should leave its answers unchanged as far as the decoded frames determine them, provided the evidence needed to answer is still observed. We call this *resampling invariance*. It is the weaker test, since an answer read from the frames that bracket an event moves with those frames whether or not the model uses time, and resampling can drop the event a question asks about, so evidence coverage is reported with every resampling comparison. Retiming is realized in two ways. In the synthetic training and evaluation data (appendix B), a scene is rendered once and attached to several timings, so retiming is built into the data. At evaluation time we instead keep the decoded frames and tell every carrier a uniform schedule with the same first and last observation time as the true one, which preserves the total span and removes only the per-observation timing; we call this *uniform retiming*. A model that reads timing should change its answers under uniform retiming, and a model that reads only the index should not. This control is used throughout section 4.

3 Tensor Clock

Tensor Clock takes the three quantities of equation (1) and routes each to the operation whose meaning it matches: the observation time t_k to the positions that attention compares, the elapsed time $\Delta t_k^{\text{state}}$ to the decay that ages the recurrent state, and both to the values through which each block is written. We first locate these operations in the backbone, then specify what each receives, which recurrent heads are modulated, and how the model is trained so that it must use them.

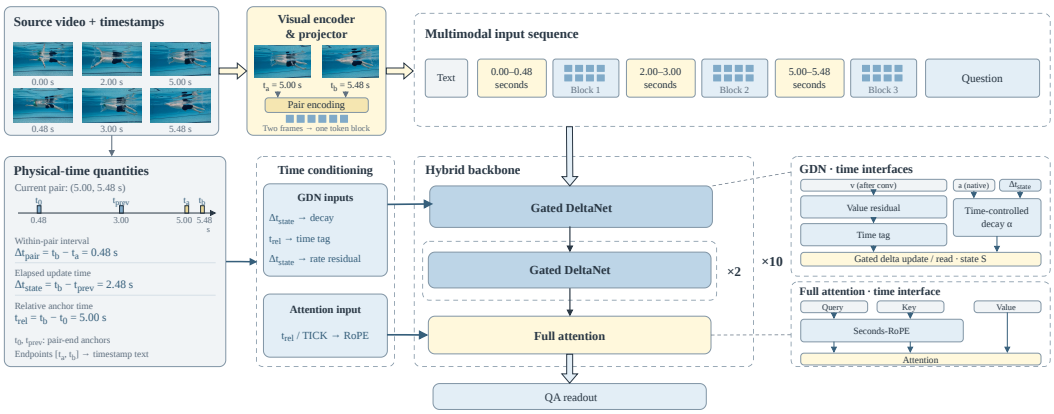


Figure 2: Tensor Clock. Each frame pair becomes one token block, preceded by its timestamp text. From the pair's source times Tensor Clock derives the observation time $t_k - t_0$ (relative anchor time in the figure) and the elapsed time $\Delta t_k^{\text{state}}$ since the previous pair, and routes them into the hybrid backbone: $\Delta t_k^{\text{state}}$ sets the decay of the selected GDN channels and scales the value residual, and $t_k - t_0$ is added as a value feature (time tag) and sets the seconds-based rotary position of attention. The within-pair interval Δt_k^{pair} reaches the model only through the text. The frozen encoder and the text are shared with the control of section 4.1, which keeps every native operation.

3.1 Backbone and observation blocks

Qwen3.6-35B-A3B encodes video with a 27-layer vision transformer whose patches span 16×16 pixels and two consecutive frames, followed by a 2×2 spatial merger. The two-frame patch is why the language model sees frame pairs: one observation block of section 2 is exactly one temporal patch, and the merger leaves it as a grid of tokens that each cover a 32×32 pixel cell. The 40-layer language backbone interleaves 10 full-attention layers with 30 Gated DeltaNet (GDN) layers. Each GDN layer keeps, for each of its 32 value heads, a matrix state S that is updated at every token i from the token's key k_i, value v_i, write gate $\beta_i \in (0, 1)$, and log decay $g_i \leq 0$, and is read with the query q_i:

$$S_i = e^{g_i}\, S_{i-1}\big(I - \beta_i k_i k_i^\top\big) + \beta_i v_i k_i^\top, \qquad o_i = S_i q_i. \tag{2}$$

The factor e^{g_i} decays the whole state and the bracket is the delta-rule write, which replaces what was stored under key k_i by v_i. All four quantities are produced by gates from the token itself, so how much the state forgets between two tokens is a function of content, not of time. We call one layer–value-head pair a *recurrent channel*, giving $30 \times 32 = 960$ channels. These are the three places where time can enter: the rotary positions read by attention, the decay gate of each channel, and the value projections through which both attention and GDN write.

3.2 What each operation receives

Every block keeps the source timestamps $(t_{k,a}, t_{k,b})$ of its two frames, and its prompt carries them as the text <t_a–t_b `seconds`>, so the timestamp text is a carrier in every model we train. The tensor operations receive the completion time $t_k = t_{k,b}$ and the elapsed time $\Delta t_k^{\text{state}}$ of equation (1). We anchor a block at its completion rather than at its midpoint so that changing the within-pair interval does not also move the block on the timeline. Rotary positions and value features use the offset $t_k - t_0$ from the first block, so that the position origin does not depend on where a clip starts in its source video, while text and answer labels use the source-video origin. The first block has no predecessor and receives $\Delta t_0^{\text{state}} = \Delta t_{\text{ref}} = 1\,\text{s}$, the reference interval at which the elapsed-time decay defined below equals the native budget. The within-pair interval Δt_k^{pair} reaches the model only through the text; no quantity is reconstructed from a frame index and a nominal frame rate.

3.3 Selecting the channels that carry elapsed time

Elapsed time should age a memory that actually spans observations; a channel that forgets within one block has nothing for elapsed time to act on. We therefore measure each channel's native

memory horizon on the pretrained model and modulate only the channels whose horizon exceeds one block. Let $g_{i,\ell h} \leq 0$ be the native log decay at token i for layer ℓ and value head h. We run the unadapted model once on a calibration clip and, at its final token T, count how many tokens back the accumulated decay still leaves half of the state:

$$L_{\ell h} = \max\left\{ n : \sum_{i=T-n+1}^{T} g_{i,\ell h} \geq -\log 2 \right\}, \qquad m_{\ell h} = \mathbf{1}[L_{\ell h} > N_{\mathrm{ref}}], \tag{3}$$

where $N_{\mathrm{ref}} = 153$ is the number of visual tokens in one block at the calibration resolution and $m_{\ell h}$ is the selection mask. The rule selects 207 of the 960 channels (21.6%). Figure 6 shows where they sit: selected channels appear in every GDN layer, most densely in the first and the last few. The mask is computed once and held fixed across training and evaluation resolutions. $L_{\ell h}$ accounts for the multiplicative decay only; delta-rule writes can change the state faster, so the horizon is a property of the gate rather than of everything the state retains. The calibration clip and the distribution of horizons are given in appendix E.

3.4 Elapsed-time decay at block boundaries

Native GDN computes the log decay as $g_{i,\ell h} = -\exp(A_{\ell h})\, y^{\mathrm{base}}_{i,\ell h}$, where $A_{\ell h}$ is a learned per-channel constant and $y^{\mathrm{base}}_{i,\ell h} > 0$ is the gate output for the token. Summed over a block, the native gate spends some log-decay budget per block; we calibrate this budget once on ordinary video as $c_{\ell h}$, the median per-block sum of y^{base} at a reference resolution (appendix E). Tensor Clock makes a selected channel spend, at each block boundary, a budget proportional to the seconds elapsed instead. Let $b_i \in \{0, 1\}$ mark the first visual token of a block. The gate becomes

$$y_{i,\ell h} = (1 - \alpha m_{\ell h})\, y^{\mathrm{base}}_{i,\ell h} + \alpha m_{\ell h}\, b_i\, \frac{\Delta t^{\mathrm{state}}_{k(i)}}{\Delta t_{\mathrm{ref}}}\, c_{\ell h}, \tag{4}$$

where $k(i)$ is the block containing token i and α interpolates between the native and the timed gate. At $\alpha = 1$ a selected channel applies its whole elapsed-time budget at the first token of each block and has a neutral gate ($g = 0$) at every other token, including text and decoding tokens; unselected channels keep their native gate. Neutral decay does not freeze the state, because the delta-rule write still updates it; what becomes independent of token count is only how much the selected channels forget between two observations, which is now set by the seconds that passed.

3.5 Positions and values

Two further operations receive time. Attention compares tokens through their rotary positions, so Tensor Clock sets the temporal rotary coordinate of every visual token from $t_k - t_0$ in seconds; the relative position between two observations then equals the time between them. The value projections are where a block is written into the recurrent state, so time is also added there. For a visual token i in block $k(i)$ and channel (ℓ, h), the value entering equation (2) becomes

$$v_{i,\ell h} \leftarrow v_{i,\ell h} + b_i\, W_{\phi,\ell h}\, \phi\big(t_{k(i)} - t_0\big) + m_{\ell h}\, \frac{\Delta t^{\mathrm{state}}_{k(i)}}{\Delta t_{\mathrm{ref}}}\, W_{r,\ell h}\, v_{i,\ell h}, \tag{5}$$

where ϕ maps a time to 16 pairs of sinusoids with periods spaced logarithmically from 0.25 to 2048 s, and $W_{\phi,\ell h}$ and $W_{r,\ell h}$ are learned per-channel matrices. The first term, applied at the first token of every block in every channel, writes the observation time itself into what the model can later read back. The second, applied at every visual token of a selected channel, scales what a block writes by the time it stands for. The keys, queries, and write gate β_i are not conditioned on time. Table 1 lists every operation, the quantity it receives, and where it applies.

4 Experiments

We first establish that Tensor Clock makes a model use physical time where a text-only control and existing models do not (section 4.2), then locate the gain in a carrier and in the operations and channels that receive time (sections 4.3 and 4.4). Section 4.5 then tests the prediction of section 2 that timing matters only under irregular observation. The settings in which the learned behavior stops are reported in appendix A.

Table 1: Where time enters. A channel is a GDN layer–value-head pair; the mask selects 207 of 960 channels. The last column states whether the operation is native to the backbone or added by Tensor Clock.

Operation	Quantity	Where it applies	Origin
Timestamp text	$t_{k,a}, t_{k,b}$	prompt, every block	native
Seconds-based rotary position	$t_k - t_0$	visual tokens, attention layers	Tensor Clock
Elapsed-time decay budget	$\Delta t_k^{\text{state}}$	block-first token, selected channels	Tensor Clock
Neutral decay	—	all other tokens, selected channels	Tensor Clock
Native decay	—	unselected channels	native
Value feature, equation (5)	$t_k - t_0$	block-first token, all channels	Tensor Clock
Value residual, equation (5)	$\Delta t_k^{\text{state}}$	visual tokens, selected channels	Tensor Clock

4.1 SETUP

Tasks. Three synthetic families hold the pixels fixed while only the timing changes (construction in appendix B). The *rate comparison* task asks which of two phases of a rendered scene moves faster; its primary score is the fraction of scenes for which every timing and every answer ordering is answered correctly, which is zero for any model that ignores timing because the pixels of a scene are identical across timings. A timestamp-only subset can be answered from the timestamps alone. The *event timing* task places events on an executable trace and asks when they happened; the *collision timing* task does the same for collisions in CLEVRER scenes. Both are scored by per-video F1 at a 0.1 s tolerance; set sizes are in appendix B. Natural video is tested on Charades-STA and QVHighlights for temporal grounding and on TOC-Bench for general video ability.

Sampling plans and timing conditions. Every evaluation video comes with a frozen sampling plan, its frame indices and source timestamps, and every model compared under a plan receives the same plan, so that a comparison across models or timing conditions changes only what the model is told about time and never the pixels. *Uniform* plans decode frames at a constant stride. *Adaptive* plans place the pairs by a motion-energy measure computed over the whole decoded video without access to labels, so that pairs are dense where the frame-to-frame change is large; the within-pair interval is one source frame (appendix B). Under *true timing* every carrier receives the source timestamps; under *uniform retiming* (section 2) it receives a uniform schedule with the same first and last observation time. Retiming can be applied to the text and to the tensor operations independently.

Models. *Tensor Clock* is the model of section 3. The *text-only control* is trained on the same data, adapter targets, steps, and random initializations, sees the same timestamp text, and keeps native positions, decay gates, and values; the two match data and optimization budget but not parameter count, since Tensor Clock adds W_ϕ and W_r. Each model is trained twice with different random initializations and every reported score averages the two runs; the statistical protocol is in appendix G. The *base model* is Qwen3.6-35B-A3B without task training. We also evaluate, without training, existing models that handle time in the two ways described in section 1: Qwen2.5-VL, whose rotary positions are aligned to seconds, and Qwen3-VL, which prefixes textual timestamps.

4.2 DOES THE MODEL USE PHYSICAL TIME?

Table 2 reports every model under true timing on adaptive plans. The base model and both existing models answer the rate comparison at or near the level of a model that ignores timing, which is expected: nothing in their training ever required the same pixels to yield different answers. Qwen2.5-VL, whose positions are already aligned to seconds, does no better than Qwen3-VL, which only writes the timestamps into the text; both are also below the base model on event timing, where the output format is unfamiliar to them. The text-only control, although it is trained on exactly the counterfactual data that would reward reading the timestamps and sees them in its prompt, does not depart from zero on the counterfactual score; the timestamps are available to it, but its computation never comes to depend on them. Tensor Clock answers 95.8% of scenes correctly under every timing. On event timing the gains are smaller and consistent, 0.07 F1 on executable traces and 0.03 on collisions. General video ability is retained: on TOC-Bench, Tensor Clock scores 0.378

Table 2: Time use under true timing on adaptive plans. Rate comparison: fraction of scenes answered correctly under every timing (180 scenes) and accuracy on the timestamp-only subset (360 questions). Event and collision timing: per-video F1 at 0.1 s. The pixel-only bound is the best score attainable while ignoring timing.

	Rate comparison ↑		Timing F1 ↑	
Model	All timings	Timestamp-only	Events	Collisions
Pixel-only bound	0.000	—	—	—
Base model	0.022	0.000	0.384	0.355
Qwen2.5-VL (seconds in positions)	0.009	0.000	0.204	0.235
Qwen3-VL (timestamps in text)	0.013	0.000	0.247	0.301
Text-only control	0.000	0.239	0.492	0.440
Tensor Clock	**0.958**	**0.996**	**0.563**	**0.474**

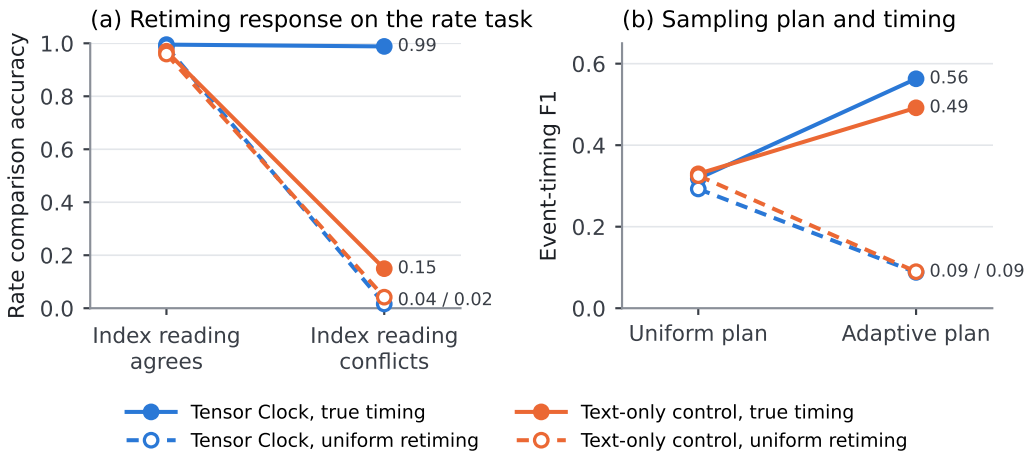


Figure 3: (a) Rate comparison accuracy on questions whose correct answer agrees or conflicts with a uniform reading of the frame index. Only Tensor Clock under true timing answers the conflicting questions, and uniform retiming removes that ability. (b) Event-timing F1 under uniform and adaptive plans. The two models agree under uniform plans; the advantage of Tensor Clock appears under adaptive plans, and retiming collapses both, since event time is read from the timestamps.

against 0.387 for the base model, a difference within noise, and abstains on about 1% of questions (appendix D).

The two tests of section 2. Figure 3a and table 9 (appendix G) report the two tests directly. Retiming response is measured on the rate questions whose correct answer differs from the answer a uniform reading of the frame index would give, so that a model can be right only by responding to the timing; the base model and the control answer these at or below chance, and Tensor Clock answers them as accurately as the others. Under uniform retiming Tensor Clock falls to zero on exactly these questions, and only there (figure 3a), which is the retiming response in the other direction. Resampling invariance is the fraction of answers that stay unchanged when the source frames are resampled while the evidence remains covered. Here the three models are close, between 0.32 and 0.44, and Tensor Clock is no more invariant than the control: this test does not separate the models. Tensor Clock therefore passes the retiming test and not the invariance test; its event answers move with the frames that bracket the event, as the control's do.

4.3 Which carrier does the trained model read?

The gain in table 2 could come from the tensor operations or from the text that both models share. To separate them we intervene on the trained Tensor Clock model at inference, retiming the text and the tensor operations independently while the pixels and weights stay fixed (table 3). Rate accuracy

Table 3: Carrier intervention on the trained Tensor Clock model, adaptive plans. Each column states which carriers receive true timing; the others are uniformly retimed. Pixels and weights are fixed.

	Text and tensors	Text only	Tensors only	Neither
Rate, all timings ↑	**0.958**	0.044	0.776	0.000
Rate, timestamp-only ↑	**0.996**	0.223	0.925	0.000
Event timing F1 ↑	**0.563**	0.534	0.112	0.088
Collision timing F1 ↑	**0.474**	0.452	0.202	0.196

Table 4: Timing F1 ↑ under sampling plan and timing condition. Uniform and adaptive plans use the same frame budget; retimed cells apply uniform retiming to every carrier.

		Uniform plan		Adaptive plan	
Task	Model	True	Retimed	True	Retimed
Events	Text-only control	**0.330**	0.325	0.492	0.090
	Tensor Clock	0.318	0.293	**0.563**	0.088
Collisions	Text-only control	0.491	0.384	0.440	0.188
	Tensor Clock	**0.518**	0.412	**0.474**	0.196

follows the tensor operations: it stays at 0.78 when only the tensors keep true timing and falls to 0.04 when only the text does, although the timestamps in the prompt are intact. Event timing follows the text: F1 is 0.53 when only the text keeps true timing, close to the 0.56 of the untouched model, and 0.11 when only the tensors do.

4.4 Which operations and which channels matter?

Table 8 in appendix F removes each part of Tensor Clock in turn, with training otherwise identical: each of the three operations that receive time, the measured channel selection against a random mask of the same size and against all channels, the counterfactual training data, and the concentration of the decay budget at the block boundary. Removing any one operation lowers rate accuracy, and the elapsed-time decay matters most: without it the counterfactual score falls from 0.96 to 0.43, whereas removing the positions or the value terms leaves 0.77. The measured mask is better than a random mask of the same size (0.96 against 0.86), and applying elapsed-time decay to every channel is worse still (0.61) and is the only variant that also hurts event timing (F1 0.41, below the control), since it removes content-dependent forgetting from the channels that need it. Trained on ordinary timestamped data, the same mechanism learns almost nothing about time (0.04): the counterfactual data, not the mechanism alone, is what forces the model to read it. Spreading the decay budget over the tokens of a block instead of the boundary costs 0.08.

4.5 When does timing matter?

Section 2 predicts that timing is redundant with the block index under uniform sampling and becomes informative only when observations are irregular. Table 4 and figure 3b cross the two sampling plans with the two timing conditions for both trained models. Under uniform plans the two models agree, on events and on collisions, and retiming the events task changes little, because the uniform schedule the model is told is nearly the true one. Under adaptive plans the picture changes in both respects: Tensor Clock exceeds the control by 0.07 F1 on events and 0.03 on collisions, and retiming collapses both models to below 0.1 on events, since the event time is read from timestamps that retiming has moved. Adaptive plans also raise event coverage on executable traces from 0.55 to 0.94, so part of the adaptive gain over uniform plans is due to observing the event at all; the model contrast within the adaptive column is not affected by coverage, because both models see the same plan.

5 DISCUSSION

The two carriers end up serving different operations, and the division is not arbitrary. Reading off when an event happened is a matter of copying a number from the context, and the timestamp text suffices for it in every model, trained or not. Comparing rates requires dividing a displacement by an interval, and no model learned that operation from text alone, even the control that was trained on data in which the division was the only way to be right. It is learned only when the interval enters the computation, through the decay that ages the recurrent state by the seconds elapsed. This also explains why seconds-aligned positions alone, as in Qwen2.5-VL, buy nothing: a position changes where an observation sits, not how the state that holds it evolves. The sampling study locates the benefit where section 2 predicts: under irregular observation.

When attention cannot see the whole video, elapsed time in the recurrent state is the only route across windows; appendix I studies this regime on an accumulation task and finds a dose-dependent benefit there but no measurable cross-window contribution on natural video.

6 RELATED WORK

Time in the input of video language models. Timestamps enter current models as text, as absolute-time tokens on visual features (Guo et al., 2024), timestamps prefixed to video patches (Bai et al., 2025a), or interleaved timestamp tokens coupled with relevance-and-coverage sampling (Yuan et al., 2025), and as position, with Qwen2.5-VL aligning its rotary coordinates to absolute time (Bai et al., 2025b). Our contribution is not timestamp injection but the use of observation intervals: under uniform sampling the block index reconstructs source time, so evaluation must change the timing independently of the pixels, which is what counterfactual schedules and uniform retiming do. Playback-speed estimation (Wu et al., 2026) infers a temporal scale from content; we instead supply source timing and ask how a model uses it.

Benchmarks. Our irregular event-time task derives from the executable-trace scoring of the Low Frequency Trap (Baskar et al., 2026). QuantiPhy asks for numerical physical quantities (Puyin et al., 2026), and recent suites probe temporal consistency, compositionality, causality, tracking, and situated awareness (Chen et al., 2026; Li et al., 2026a; Zhang et al., 2026; Liu & Lee, 2026; Li et al., 2026b; Cao et al., 2026); we measure their within-pair displacements and audit the source timing of four of them (appendix C). Charades-STA and QVHighlights are used unmodified (Gao et al., 2017; Lei et al., 2021). None of these benchmarks changes the temporal schedule independently of the decoded frames.

Time in recurrent dynamics. Conditioning recurrent dynamics on elapsed time predates video language models: Phased LSTM gates updates with an oscillating clock (Neil et al., 2016), ODE-RNN and liquid time-constant networks evolve state continuously between observations (Rubanova et al., 2019; Hasani et al., 2021), and continuous-time state-space models are built for irregular sampling (Ansari et al., 2023; Nguyen et al., 2022). These train time into the dynamics from scratch; we take a pretrained backbone whose gated linear recurrence (Yang et al., 2025) is interleaved with attention (Qwen Team, 2026) and condition a measured subset of its channels on elapsed time at block boundaries. External memories for long video (Song et al., 2024) are addressable by content or location (Sukbaatar et al., 2015), not by time.

7 CONCLUSION

Once a video is sampled under a fixed frame budget, frame order no longer determines elapsed time, and a model that is only shown the timestamps does not learn to use them. Tensor Clock routes the observation time and the elapsed time between observations into the positions, the recurrent decay, and the values of a hybrid backbone, and with counterfactual training it makes rate reasoning under irregular observation possible where a control given identical timestamps in text never departs from zero. The recurrent decay carries most of the effect, the measured channel selection outperforms random and full selection, and the counterfactual data is what forces time to be read. Using the within-pair interval alone and natural-video grounding remain open.

REFERENCES

Abdul Fatir Ansari, Alvin Heng, Andre Lim, and Harold Soh. Neural continuous-discrete state space models for irregularly-sampled time series. In *International Conference on Machine Learning*, 2023. URL https://arxiv.org/abs/2301.11308.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, Mei Li, Kaixin Li, Zicheng Lin, Junyang Lin, Xuejing Liu, Jiawei Liu, Chenglong Liu, Yang Liu, Dayiheng Liu, Shixuan Liu, Dunjie Lu, Ruilin Luo, Chenxu Lv, Rui Men, Lingchen Meng, Xuancheng Ren, Xingzhang Ren, Sibo Song, Yuchong Sun, Jun Tang, Jianhong Tu, Jianqiang Wan, Peng Wang, Pengfei Wang, Qiuyue Wang, Yuxuan Wang, Tianbao Xie, Yiheng Xu, Haiyang Xu, Jin Xu, Zhibo Yang, Mingkun Yang, Jianxin Yang, An Yang, Bowen Yu, Fei Zhang, Hang Zhang, Xi Zhang, Bo Zheng, Humen Zhong, Jingren Zhou, Fan Zhou, Jing Zhou, Yuanzhi Zhu, and Ke Zhu. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025a. URL https://arxiv.org/abs/2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025b. URL https://arxiv.org/abs/2502.13923.

Sarvesh Baskar, Zikui Cai, Shayan Shabihi, Anirudh Satheesh, Muhammad R. Islam, Udari Madhushani Sehwag, Tom Goldstein, and Furong Huang. The low frequency trap: Video language models fail at simple event bookkeeping. *arXiv preprint arXiv:2608.06361*, 2026. URL https://arxiv.org/abs/2608.06361.

Tri Cao, Khoi Le, Thong Nguyen, Cong-Duy Nguyen, Quynh Vo, Anh Tuan Luu, Chunyan Miao, See-Kiong Ng, Shuicheng Yan, and Bryan Hooi. Tracking the truth: Object-centric spatio-temporal monitoring for video large language models. *arXiv preprint arXiv:2605.08974*, 2026. URL https://arxiv.org/abs/2605.08974. Withdrawn by the authors on arXiv (evaluation and aggregation errors); cited only as the source of the STEMO-Bench measurements we report.

Junzhe Chen, Siyuan Meng, Yuxi Chen, Man Zhao, Wenyao Gui, and Xiaojie Guo. TOC-Bench: A temporal object consistency benchmark for video large language models. *arXiv preprint arXiv:2605.09904*, 2026. URL https://arxiv.org/abs/2605.09904.

Jiyang Gao, Chen Sun, Zhenheng Yang, and Ram Nevatia. TALL: Temporal activity localization via language query. In *Proceedings of the IEEE International Conference on Computer Vision*, 2017. URL https://arxiv.org/abs/1705.02101.

Yongxin Guo, Jingyu Liu, Mingda Li, Dingxin Cheng, Xiaoying Tang, Dianbo Sui, Qingbin Liu, Xi Chen, and Kevin Zhao. VTG-LLM: Integrating timestamp knowledge into video LLMs for enhanced video temporal grounding. *arXiv preprint arXiv:2405.13382*, 2024. URL https://arxiv.org/abs/2405.13382.

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2006.04439.

Jie Lei, Tamara L. Berg, and Mohit Bansal. QVHighlights: Detecting moments and highlights in videos via natural language queries. In *Advances in Neural Information Processing Systems*, 2021. URL https://arxiv.org/abs/2107.09609.

Baiqi Li, Kangyi Zhao, Ce Zhang, Chancharik Mitra, Jean de Dieu Nyandwi, and Gedas Bertasius. Timeblind: A spatio-temporal compositionality benchmark for video llms. *arXiv preprint arXiv:2602.00288*, 2026a. URL https://arxiv.org/abs/2602.00288.

Chuan Li, Rilyn Han, Joy Hsu, Yongyuan Liang, Rajiv Dhawan, Jiajun Wu, Ming-Hsuan Yang, and Xin Eric Wang. SAW-Bench: Learning situated awareness in the real world. *arXiv preprint arXiv:2602.16682*, 2026b. URL https://arxiv.org/abs/2602.16682.

Tiedong Liu and Wee Sun Lee. Can vision-language models solve the shell game? *arXiv preprint arXiv:2603.08436*, 2026. URL https://arxiv.org/abs/2603.08436.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased LSTM: Accelerating recurrent network training for long or event-based sequences. In *Advances in Neural Information Processing Systems*, 2016. URL https://arxiv.org/abs/1610.09513.

Eric Nguyen, Karan Goel, Albert Gu, Gordon W. Downs, Preey Shah, Tri Dao, Stephen A. Baccus, and Christopher Ré. S4ND: Modeling images and videos as multidimensional signals using state spaces. In *Advances in Neural Information Processing Systems*, 2022. URL https://arxiv.org/abs/2210.06583.

Li Puyin, Tiange Xiang, Ella Mao, Shirley Wei, Xinye Chen, Adnan Masood, Li Fei-Fei, and Ehsan Adeli. QUANTIPHY: A quantitative benchmark evaluating physical reasoning abilities of vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 33174–33184, 2026. URL https://arxiv.org/abs/2512.19526.

Qwen Team. Qwen3.6-35B-A3B: Model card and configuration. Hugging Face model repository, 2026. URL https://huggingface.co/Qwen/Qwen3.6-35B-A3B. Accessed September 25, 2026.

Yulia Rubanova, Ricky T. Q. Chen, and David K. Duvenaud. Latent ODEs for irregularly-sampled time series. In *Advances in Neural Information Processing Systems*, 2019. URL https://arxiv.org/abs/1907.03907.

Enxin Song, Wenhao Chai, Guanhong Wang, Yucheng Zhang, Haoyang Zhou, Feiyang Wu, Haozhe Chi, Xun Guo, Tian Ye, Yanting Zhang, Yan Lu, Jenq-Neng Hwang, and Gaoang Wang. MovieChat: From dense token to sparse memory for long video understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024. URL https://arxiv.org/abs/2307.16449.

Sainbayar Sukbaatar, Arthur Szlam, Jason Weston, and Rob Fergus. End-to-end memory networks. In *Advances in Neural Information Processing Systems*, 2015. URL https://arxiv.org/abs/1503.08895.

Yen-Siang Wu, Rundong Luo, Jingsen Zhu, Tao Tu, Ali Farhadi, Matthew Wallingford, Yu-Chiang Frank Wang, Steve Marschner, and Wei-Chiu Ma. Seeing fast and slow: Learning the flow of time in videos. *arXiv preprint arXiv:2604.21931*, 2026. URL https://arxiv.org/abs/2604.21931.

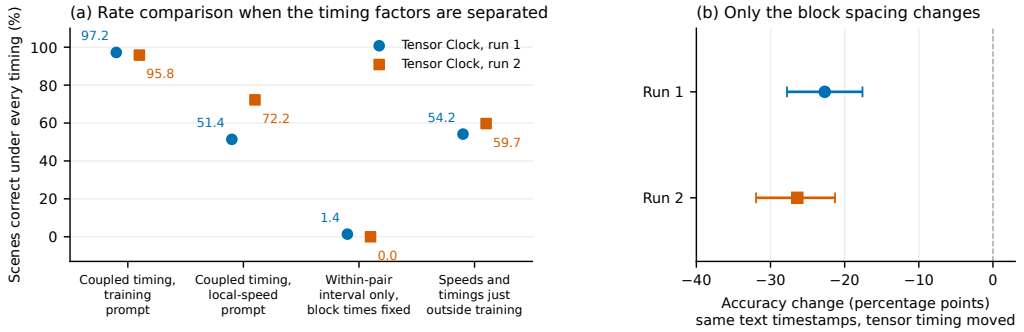
Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2412.06464.

Chao Yuan, Yang Yang, Yehui Yang, and Zach Cheng. DATE: Dynamic absolute time enhancement for long video understanding. *arXiv preprint arXiv:2509.09263*, 2025. URL https://arxiv.org/abs/2509.09263.

Mingfang Zhang, Jingjing Pan, Ashutosh Kumar, Rajat Saini, Mustafa Erdogan, Hsuan-Kung Yang, Caixin Kang, Yifei Huang, Yoichi Sato, and Quan Kong. CaST-Bench: Benchmarking causal chain-grounded spatio-temporal reasoning for video question answering. *arXiv preprint arXiv:2605.23216*, 2026. URL https://arxiv.org/abs/2605.23216. CVPR 2026.

A LIMITATIONS

The learned behavior has boundaries. The rate task couples the within-pair interval with the block timing, and when the two are varied separately neither is used on its own: with block timing fixed and only the within-pair interval changed, Tensor Clock answers at most 1 of 72 scenes correctly under every timing, and when only the waiting time between blocks changes, which is irrelevant to the question, accuracy drops by 23 to 26 points (figure 4). Timings outside the training range are untested. On natural-video temporal grounding the model is indistinguishable from the control: on Charades-STA and QVHighlights under three sampling plans, every paired difference in recall spans zero (appendix H), so the timing that decides the synthetic tasks is not what limits models on those benchmarks.



(a) Point estimates over 72 scenes per setting; zero observed successes is not a population bound.

(b) Paired 95% scene-bootstrap intervals, post hoc; a different metric from (a).

Figure 4: Where the learned behavior stops. (a) Fraction of scenes answered correctly under every timing, 72 scenes per setting, for both training runs of Tensor Clock: the coupled timing of training with the training prompt; the same timing with a prompt stating that only the speed within each pair counts; the within-pair interval varied alone with block times fixed; and speeds and timings just outside the training range. (b) Change in accuracy when only the spacing between blocks is moved in the tensor timing while the text timestamps stay true, paired 95% scene-bootstrap intervals.

The evidence covers one backbone with a frozen vision encoder and synthetic or derived timing tasks. Tensor Clock adds about 20M value parameters, and no control with the same parameter count was trained. The channel mask comes from a single calibration clip. The natural-video pilot gives the text a single midpoint timestamp per block, whereas training used pair-range text (appendix H). Source timestamps require a known correspondence to capture time, which editing and slow motion break, and metric velocity additionally requires spatial calibration. The answer-token weighting that the rate family required induces about 1% abstention on natural video.

B DATA, TRAINING, AND PREREGISTRATION

Training mixture. The frozen mixture contains 6 740 rows from 4 660 videos: rate2d v5b training scenes (1 620 rows, 24.0%), LFT-irregular training (1 920, 28.5%), CLEVRER-derived training (1 200, 17.8%), T-LEAK (1 500, 22.3%), and a held-out retention set (500, 7.4%). The text carrier presents true range text throughout; for the event-time families, pair placement is uniform or motion-adaptive with equal probability per video, and each pair's interval is one source frame with probability one half and otherwise log-uniform between one source frame and 0.5 s (executable traces) or 0.2 s (collisions). One epoch is 211 steps at 32 samples per step. Answer tokens of the rate family are supervised with weight 3; ledger-style answers of the event-time families are supervised uniformly. A file-hash contamination check between every training and evaluation set passed.

Evaluation set sizes. The rate comparison set has 300 scenes with 540 timings and 1 620 questions; its primary score is computed over the 180 scenes with several timings, and the timestamp-

only subset has 360 questions. Event timing uses 1 120 videos at 64 frames; collision timing uses 400 CLEVRER scenes at 32 frames.

Counterfactual construction. Routing time into the computation does not by itself make the model use it: on an ordinary timestamped dataset, pixels and labels are correlated, so the model can fit the labels without reading any carrier. We therefore construct training and evaluation sets in which the pixels are held fixed and only the timing changes, so that the same frames must yield different answers. In the rate family, a scene is rendered once and paired with three timings of equal total duration, so that the same decoded frames imply three different rate orderings; answers are balanced within each timing, which caps any strategy that ignores timing at the pixel-only bound (5/9 over the pool, and 0 for answering all timings of a scene correctly). In the event-time family, events are placed on an executable trace and frames are sampled irregularly around them, so that the reported time of an event depends on the source times of the frames that bracket it. A synthetic accumulation task, which requires carrying a count across observations, and a held-out retention set complete the mix.

Rate family construction. Each scene fixes its rendered motion across temporal variants. Schedules drawn from three patterns retain the total duration (24 s); questions compare rates in the first and second phases. The evaluation pool contains 300 scenes, of which 180 have multiple retained schedules. Answers are balanced within each schedule. A shortcut suite verifies that displacement-ratio lookup, schedule lookup, and constant answers cannot exceed the pixel-only bound (5/9 over the pool, 1/3 on the timestamp-only subset).

Event timing on executable traces (LFT-irregular). Videos are rendered from an executable program whose trace supplies the event times: a bouncing ball whose wall contacts are the events, a blinking object whose pulses are the events, or a state machine whose transitions are the events. Each video lasts 24 s at 30 frames per second. The gaps between consecutive events are drawn log-normally with a coefficient of variation $\mathrm{cv} \in \{0, 1, 2, 3\}$, rescaled so that the active span is the same at every cv, and floored at 0.1 s for the ball and at one source frame otherwise; the onset of the active span is uniform within the idle time. The evaluation set has 1 120 videos: 960 ball videos crossing 8, 10, or 12 events, event rates from 0.5 to 4 per second, the four cv values, and ten random draws, plus 160 blinking and state-machine videos. The training set has 1 920 videos from disjoint random draws with 6 to 12 events; the rate of 3.5 per second is held out of training. Every plan decodes 64 frames as 32 pairs, and the answer lists the time of every event.

Collision timing (CLEVRER-derived). Events are the collisions annotated in CLEVRER, timed at the annotated frame at 25 frames per second; collisions annotated in the same frame are merged into one event. A scene is kept if it has two to five collisions, consecutive collisions are at least 0.24 s apart, and no collision lies within 0.3 s of either end of the 5.12 s video. Evaluation uses the first 400 qualifying scenes of the CLEVRER training split (239 with two collisions, 153 with three, 8 with four); training uses 1 200 disjoint qualifying scenes. Every plan decodes 32 frames as 16 pairs, and the maximum training pair interval of 0.2 s stays below the minimum collision gap.

Adaptive plans. The motion-energy measure is the mean absolute difference between consecutive 64×36 grayscale thumbnails of the fully decoded video. Four pairs are placed at uniform positions as a reserve; the starts of the remaining pairs are placed at the quantiles $(k + \frac{1}{2})/K$ of the cumulative motion energy, so that pairs are dense where the frame-to-frame change is large. No threshold or peak detection is involved. Pairs are deduplicated and kept disjoint and in order, and the budget is filled by splitting the largest gaps. The within-pair interval is one source frame ($1/30$ s on executable traces, 0.04 s on collisions). The adaptive plans cover 94% of events on executable traces against 55% for uniform plans; on collisions both cover nearly all events (99% and 100%).

Accumulation task (T-LEAK). Two balls, red and blue, flash yellow at irregular moments; each flash adds one unit to that ball's quantity, which then decays exponentially with a time constant of 12 s, and the question asks which ball has more remaining at the end. The source video runs at 8 frames per second and each block is an adjacent frame pair, so the within-pair interval is 0.125 s. A video has 24 or 32 blocks, 8, 16, or 24 flashes split evenly between the balls, and lasts 16 to 96 s; the gaps between block centers lie in $[0.25, 8]$ s and alternate between three to six dense regions, log-

uniform in $[0.25, 0.5]$ s, and sparse regions in $[2, 8]$ s. Items are kept only when the two accumulated quantities differ by a ratio of at least 1.15, and are labeled by whether the answer under true time agrees or conflicts with the answer under block counting. The mixture holds 750 videos, each in both option orders.

Training configuration. Both matched arms start from Qwen3.6-35B-A3B with the vision transformer and merger frozen and train the same low-rank adapter targets (rank 16), plus W_ϕ and W_r for Tensor Clock, for two epochs: 422 steps of 32 samples, learning rate 10^{-5} with a cosine schedule, frames decoded at 224^2 pixels. The decay interpolation α ramps linearly from 0 to 1 over the first 211 steps, so that adaptation starts from the native behavior. Answer tokens carry loss weight 3, without which the rate family did not depart from chance. The main checkpoint is the final one; no checkpoint was selected on evaluation results. On TOC-Bench, retimed and true cells are bit-identical when every carrier receives the same timing, so retiming changes only what a carrier is told.

Preregistration. Protocols were frozen before results were inspected for: the main training run (data proportions, endpoints, pass criteria, final-checkpoint rule); the sampling-by-clock supplement (six contrasts, Holm correction, cluster definitions); the four-cell ablation (cell definitions, sixteen primary contrasts, the two preset asymmetries); and the matched-budget retraining (four arms, four primary contrasts, non-inferiority margin 0.02). The natural-video pilot froze its 128-video selection by hash before any inference.

C Readable-Band Measurements

This appendix gives the measurements behind the readable band of section 2. They characterize the frozen vision encoder shared by every model we train and motivate choosing the pair interval per video.

Why a fixed budget forces the interval to vary. Within one block, the evidence for motion is the displacement d of image content between its two frames. For approximately constant velocity v, in source-image pixels per second, and resize scale s, this displacement is $d \approx v\,\Delta t^{\mathrm{pair}}\,s$, so at a fixed velocity the interval sets the displacement. The frozen vision encoder of our backbone reads displacement only in a limited range: a linear probe for motion direction on its output features is at the level of random weights for 1-pixel displacements, rises through 4 and 8 pixels, saturates at 16 to 32 pixels, and drops at 64 pixels, where the displacement exceeds the width of one feature cell; we call $[8, 32]$ pixels the *readable band*. Under the usual 64-frame uniform sampling, the median within-pair displacement of four of seven natural-video benchmarks falls below this band, and the two benchmarks most concerned with measuring motion, QuantiPhy and TimeBlind, are the furthest below (table 6). The band spans a factor of four, so with a fixed budget the interval has to be chosen per video or per segment, longer where motion is slow and shorter where it is fast. Adaptive frame selection, variable frame rate, dropped frames, and event-triggered capture push in the same direction.

Probe design. We render two-frame clips in which a textured patch moves by a controlled displacement $d \in \{1, 2, 4, 8, 16, 32, 64\}$ pixels between the two frames of a block. The vision transformer (ViT) of Qwen3.6-35B-A3B uses 16×16 pixel patches followed by a 2×2 spatial merger, so each feature that reaches the language model covers a 32×32 pixel cell; we call these features the *interface*. On them we fit a linear eight-way motion-direction classifier and report macro accuracy, for which chance is 0.125. To separate what pretraining contributes from what any fixed projection of the pixels would give, we fit the same classifier on features from the same architecture with randomly initialized weights (the floor). Table 5 and figure 5 report the sweep. At 1 pixel the trained encoder is indistinguishable from the floor; accuracy rises through 4 and 8 pixels, saturates at 16 to 32 pixels, and declines at 64 pixels, where the displacement exceeds the width of a cell. We refer to $[8, 32]$ pixels as the readable band and to $[16, 32]$ as its peak; the band endpoints differ by a factor of four and the peak by two, which bounds how far the interval can vary at a fixed velocity before readability drops.

Table 5: Motion-direction readability versus within-pair displacement (eight-way macro accuracy, chance 0.125; 32-pixel cells). Interface is the feature the language model receives; the floor is the same architecture with random weights; L12 and L27 are encoder layers pooled to the same granularity. Scene-clustered bootstrap intervals are described under Sweep.

Displacement (px)	1	2	4	8	16	32	64
Interface (trained)	0.172	0.306	0.468	0.707	0.910	0.914	0.640
Floor (random weights)	0.194	0.268	0.386	0.499	0.695	0.704	0.523
Encoder L27, pooled	0.168	0.315	0.457	0.727	0.928	0.944	0.719
Encoder L12, pooled	0.231	0.466	0.677	0.875	0.947	0.947	0.747

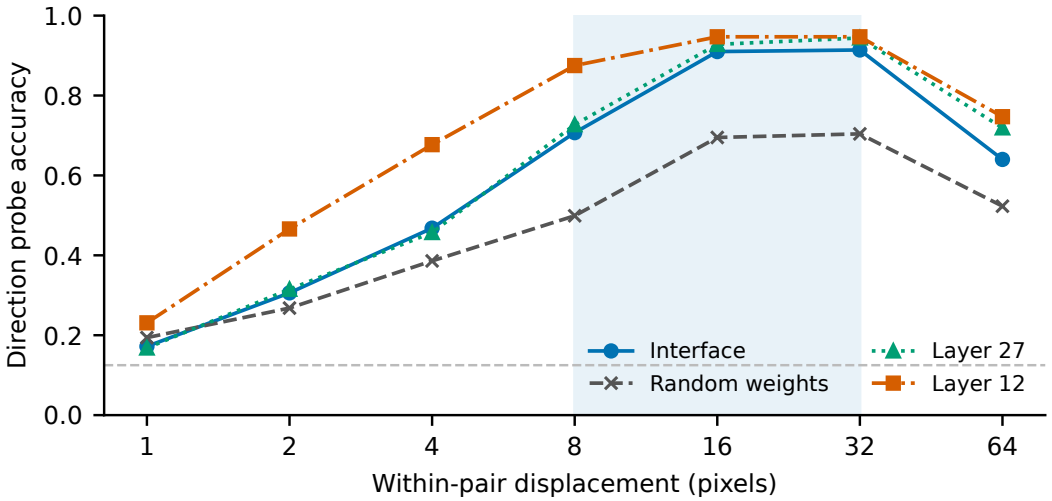


Figure 5: Direction accuracy of the frozen interface features versus within-pair displacement. The shaded region is the 8–32 pixel readable band. Points reproduce table 5.

Sweep. Two runs with different clip lengths ($T = 6$ and $T = 12$) agree point for point on the overlapping displacements, so clip length does not change the shape. Trained-minus-floor gaps at the interface are -0.022, $+0.038$, $+0.081$, $+0.209$, $+0.215$, $+0.210$, $+0.118$ for 1 to 64 pixels; all but the first have scene-clustered bootstrap lower bounds above zero. The floor itself rises with displacement, so the gap understates readability at large displacements; we use raw accuracy to judge whether the interface can read direction and the gap to judge whether pretraining contributes.

Readability across encoder depth. Whether the band is a property of the pixels at this resolution or of the encoder matters for how it should be handled, so we probe intermediate layers pooled to the same 32-pixel granularity. Layer 12 reads small displacements better than layer 27 and the interface: its advantage over the interface is about 0.17 to 0.21 at 2 to 8 pixels, while layer 27 and the interface are close (table 5). The reduction in linear decodability is therefore located in the later encoder layers. Concatenating pooled layer-12 (or layer-6) features to the interface recovers $+0.17$ to $+0.21$ at 2 to 8 pixels, within $+0.012$ of the pooled mid-layer features alone, so the gain comes from depth rather than from added dimensionality; layer 12 is the better single choice, being slightly stronger at 64 pixels. The timing experiments do not use this bypass.

Displacements in sampled natural videos. In four of these benchmarks (TimeBlind, TOC-Bench, CaST-Bench, STEMO-Bench) we also read the source frame rate of every video: it is constant to within 10^{-5} for the large majority of videos, so decoding at a fixed index stride yields exactly the uniform schedule of section 2. To see where standard sampling places natural video relative to the band, we decoded 64 frames per video at 224^2 pixels for seven natural-video benchmarks, estimated the dominant local displacement inside each pair, and took the median over about ten videos per set (table 6). Four sets fall below the band (QuantiPhy 0.0, TimeBlind 1.6, CaST-Bench 4.5, TOC-Bench 7.4 pixels) and three inside it (STEMO-Bench 12.0, VET-Bench 12.8, SAW-Bench

16.6 pixels); none lies above. The two sets most concerned with measuring motion, QuantiPhy and TimeBlind, are the two furthest below. The same fixed budget also gives very different pair intervals across sets, from 0.03 to 4.2 s, so a reading of displacement that ignores the interval would already be wrong by two orders of magnitude between benchmarks. For QuantiPhy some sampled pairs even repeat a source frame, because the clips have fewer source frames than the budget. These are order-of-magnitude estimates from small samples. The local-displacement estimator takes the median displacement of the 10% of 64-pixel windows with the largest temporal energy, so it reflects the strongest local motion in the frame rather than a specific object; its measurable range is ± 32 pixels and no reading approached it. QuantiPhy clips have a median of 69 source frames against a 64-frame budget, so 11 of 24 sampled videos have a pair interval shorter than one source frame interval.

Table 6: Within-pair displacement on natural benchmarks (64 frames per video, 224^2 pixels, about ten videos per set; medians of the dominant local displacement). Band is $[8, 32]$ pixels.

Benchmark	Sampled span (s)	Δt^{pair} (s)	Local displacement p50 (px)	Relative to band
QuantiPhy	2.0	0.032	0.0	far below
TimeBlind	9.3	0.145	1.6	below
CaST-Bench	13.1	0.204	4.5	below
TOC-Bench	30.1	0.470	7.4	at lower edge
STEMO-Bench	268.7	4.199	12.0	inside
VET-Bench	12.2	0.191	12.8	inside
SAW-Bench	36.1	0.564	16.6	inside

D Output Format and Abstention Diagnostics

Table 7: Output quality by cell (trained checkpoint, 1 024 new tokens). Truncated and malformed outputs are scored as they stand and were not removed.

Cell	Format compliance	Truncation	Explicit abstention
LFT-irregular, uniform, true	0.906	0.094	0
LFT-irregular, uniform, retimed	0.872	0.128	0
LFT-irregular, adaptive, true	0.975	0.025	0
LFT-irregular, adaptive, retimed	0.986	0.014	0
CLEVRER-derived, all four cells	1.000	0	0

TOC-Bench abstentions. Counts by training configuration: base 0; unweighted one epoch 1; weighted (3) one epoch 34; weighted two epochs 61. Forty of the 61 fall in the cross-object ordering subtask (21% of that subtask), on which the base model scores 0.508. The base model under the same pair-range input format scores 0.3874, within five items of its score under its native input format, so the drop is attributable to the trained weights and not to the input format.

E GDN Channel Selection and Calibration

The selection mask is computed once from the unadapted Qwen3.6-35B-A3B checkpoint. The calibration uses one PointOdyssey training clip at 2 frames/s and measures gates at the final token of the resulting 12 153-token sequence. The processed grid has 75 temporal blocks, each with 18×34 spatial patches before 2×2 merging, hence 153 visual tokens per block. For each of the 30 GDN layers and 32 value heads, we accumulate its native log decay backwards until the gate-only survival factor drops below one half. Equation (3) freezes the resulting mask before task adaptation.

Decay lookback (tokens)	Heads	Selected
0–10	736	no
$10 < L \leq 153$	17	no
$153 < L < 12\,153$	151	yes
At the sequence limit (12 153)	56	yes

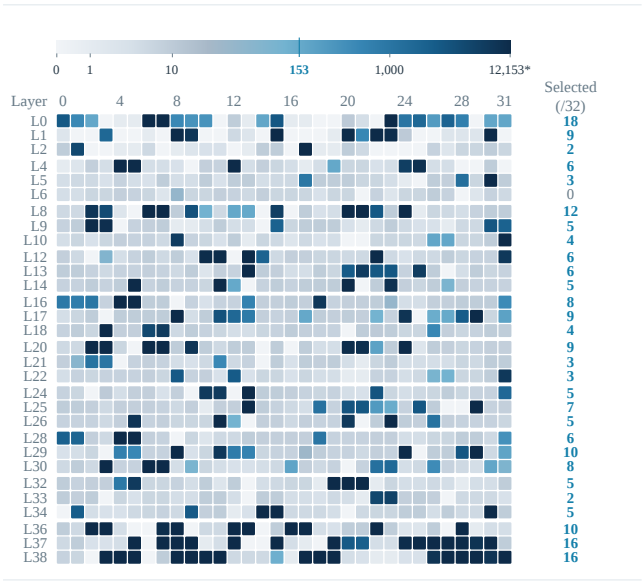


Figure 6: Channel selection by decay lookback. Each cell is one GDN layer–value-head channel, coloured by its lookback $L_{\ell h}$ on the calibration clip (log scale). The 207 channels with $L_{\ell h} > 153$ tokens, one observation block, receive elapsed-time decay; the right column counts them per layer. Rows skip the ten full-attention layers.

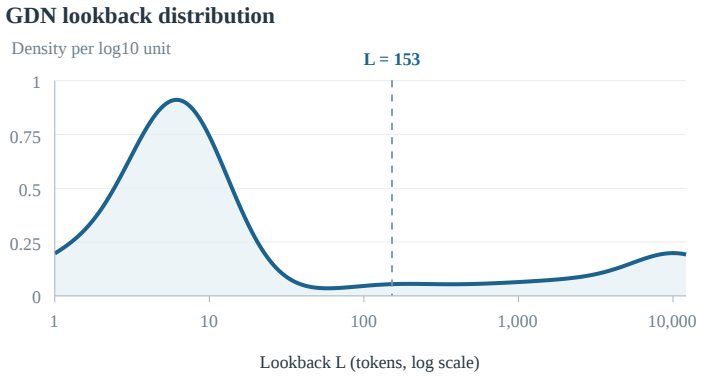


Figure 7: Distribution of decay lookback $L_{\ell h}$ over the 960 channels (density per decade). Most channels forget within ten tokens; a second mode sits at the sequence limit of 12 153 tokens, where $L_{\ell h}$ is right-censored. The dashed line is the selection threshold of one block, 153 tokens.

The last row is right-censored at the sequence length. $L_{\ell h}$ measures the multiplicative decay on one calibration clip (section 3.3), and the mask is used as a fixed design rule. A random mask of the same size and the all-channel variant are compared in table 8; a threshold sweep has not been run.

The decay budget $c_{\ell h}$ uses a separate frozen calibration: the median per-block sum of the native positive softplus gate across 12 blocks from two clips at a reference resolution of 256×512 (128 merged visual tokens per block). This gate-budget reference differs from the 153-token resolution used to choose the mask. Both artifacts are loaded unchanged during task training. The Tensor Clock runs anneal the decay interpolation over 211 steps and reach $\alpha = 1$, after which a selected channel has a neutral decay gate at every token other than the first visual token of a block (section 3.4).

F Ablation Details

The first group of table 8 asks which of the three places where time enters is needed: seconds-based positions, elapsed-time decay, or the value terms. The second group tests the channel selection of section 3.3 by applying elapsed-time decay to a random set of channels of the same size and to all channels. The third group trains on ordinary timestamped data instead of the counterfactual mixture, which tells whether the data or the mechanism is what makes the model read time, and spreads the decay budget over the tokens of a block instead of concentrating it at the boundary. Training is otherwise identical to the full model.

Table 8: Ablations. Each row changes one part of Tensor Clock; training is otherwise identical. Rate: fraction of scenes correct under all timings; F1: event timing on executable traces, adaptive plans.

Variant	Rate ↑	Event F1 ↑
Tensor Clock	**0.958**	**0.563**
without seconds-based positions	0.776	0.511
without elapsed-time decay	0.431	0.501
without value terms	0.765	0.496
decay on a random 207 channels	0.863	0.545
decay on all 960 channels	0.612	0.413
trained on ordinary timestamped data	0.042	0.508
decay budget spread over block tokens	0.881	0.555

G Time-Use Tests and Statistics

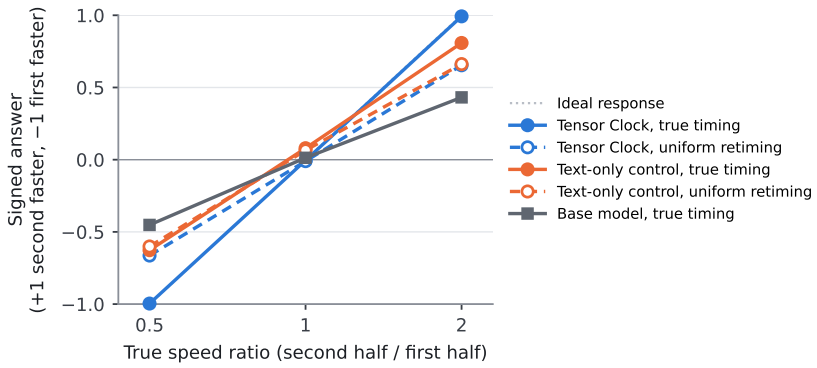


Figure 8: Response of the rate answer to the timing. For each true speed ratio that the timing implies, the signed answer averages $+1$ for "second half faster", -1 for "first half faster", and 0 for "the same". A model that reads time follows the dotted ideal line; a model that ignores it has a shallower slope, since it is right only where the index-based reading happens to agree.

Table 9 gives the two tests of time use defined in section 2 and discussed in section 4.2. Every trained model was trained twice with different random initializations, and every reported score is the average of the two runs. Intervals are paired 95% cluster-bootstrap intervals with 10 000 resamples, clustered by scene or source video; the comparison averages the two runs' scores of each test item before resampling and does not treat runs as extra test items. The four primary contrasts (rate all-timings, timestamp-only, event F1, collision F1) were preregistered with a Holm correction; all other intervals are descriptive. The direction of every primary contrast is the same in both runs.

Table 9: The two tests of time use. Retiming response: accuracy on the 720 rate questions whose correct answer differs from the index-based reading (chance 0.33). Resampling invariance: fraction of event-timing answers unchanged when the uniform plan is replaced by the adaptive plan, restricted to the 94% of questions whose evidence the adaptive plan still covers.

Model	Retiming response ↑	Resampling invariance ↑
Base model	0.236	0.324
Text-only control	0.149	0.423
Tensor Clock	**0.988**	**0.442**

H Natural-Video Grounding Pilot

We evaluate 380 Charades-STA queries and 133 QVHighlights queries from a frozen 128-clip-per-dataset subset, with 64 input frames under uniform, controlled-irregular, and adaptive plans. All models share the decoded observations and source timestamps. The matched models were trained with pair-range text; this inference comparison uses a single midpoint timestamp per block in the text while preserving the end-anchored tensor timing, and recovers an interval from the raw answer with a fixed parser.

Table 10: Natural-video grounding pilot, R@1 at IoU 0.5 (%). Columns average the two training runs; the difference is Tensor Clock minus control in percentage points with a paired 95% interval. QVHighlights uses 111 source-video clusters for 128 clips.

Dataset	Plan	Control	Tensor Clock	Difference [95% CI]
Charades-STA	uniform	75.26	74.61	-0.66 [-2.46, $+1.01$]
Charades-STA	irregular	72.76	72.50	-0.26 [-2.66, $+2.23$]
Charades-STA	adaptive	73.68	74.08	$+0.39$ [-1.74, $+2.62$]
QVHighlights	uniform	72.56	69.17	-3.38 [-8.40, $+1.47$]
QVHighlights	irregular	66.54	65.41	-1.13 [-5.56, $+3.17$]
QVHighlights	adaptive	67.67	68.42	$+0.75$ [-3.76, $+5.28$]

I Analysis: Time in the Recurrent State under Windowed Attention

Bounded attention is the second place where frame counting is substituted for time. Under chunk-isolated attention, the final query cannot attend to earlier chunks, so information from those chunks must pass through the recurrent state. Native GDN applies content-dependent decay per token; the elapsed-time decay of section 3.4 makes forgetting a function of elapsed time. This section measures what that changes: the elapsed-time decay helps on a synthetic accumulation task under windowed attention and shows no measurable effect on STEMO-Bench.

Two conditions under which the state clock is identifiable. The elapsed-time decay differs from a per-block decay only when two conditions hold together. Attention must not see the whole video, otherwise the answer can be read from visible blocks and their visible timestamps; and the observation schedule must be irregular, otherwise Δt^{state} is a constant and the two decays coincide up to a gain (section 2). Under chunked attention the text timestamps of earlier chunks are cut off together with their frames, so the text clock is no longer global; this is where the state clock has to carry the timing on its own.

A synthetic task that requires the state. T-LEAK is an accumulation task: a quantity leaks at a rate that changes at events, frames are sampled irregularly with block intervals in $[0.25, 8]$ s, and the question requires integrating over the whole video. Under window $W = 8$ with 32 blocks, an oracle that integrates perfectly over only the last chunk scores 0.663, so the task cannot be solved from the visible window. We trained five arms for 4 438 steps on this task alone, twice with different random initializations, and evaluate the fraction of items answered correctly in both orderings. The native recurrence without any Δt reaches 0.847, well above the last-chunk oracle, so the native linear state

already carries cross-window information once it is trained under windows. Conditioning decay on true Δt^{state} reaches 0.901, and the matched null that conditions the same gate on block count reaches 0.818; the paired difference is $+0.081$ [$+0.058$, $+0.106$] in one run and $+0.080$ [$+0.064$, $+0.098$] pooled over both runs, rising to $+0.170$ [$+0.108$, $+0.232$] on the subset whose evidence lies entirely outside the last chunk. Scaling the decay gain $\lambda \in \{0, 0.25, 0.5, 1\}$ gives 0.578, 0.715, 0.813, 0.901, a monotone dose response, and the effect doubles from the most regular to the most irregular quartile of block-interval variation ($+0.052$ to $+0.104$). It is retained at 91% when the per-frame text timestamps are made visible inside each chunk. The gain over the base recurrence, $+0.056$ [$+0.040$, $+0.072$] pooled, did not meet the preregistered threshold of a 0.05 lower bound and a 0.10 point estimate.

On natural video the state pathway carries nothing we can measure. We repeated the measurement on STEMO-Bench (3 920 questions, 71 videos, 64 frames as four chunks of eight blocks), using the arm with native decay and balanced accuracy: STEMO-Bench gold answers are 74.6% "yes" and arms differed in their "yes" rate by up to 8 points, so raw-accuracy differences between arms largely reflected threshold shifts, and under balanced accuracy all six preregistered arm contrasts include zero. Three conditions are compared on the same questions: the last chunk alone, the whole video under $W = 8$, and the whole video with full attention. Because the chunk mask assigns the question to the last chunk, the difference between the first two conditions is the entire contribution of the state carrying the first 24 blocks: $+0.004$ [-0.014, $+0.022$] under the realistic text condition and -0.008 [-0.029, $+0.015$] under the mechanism condition. The same 24 blocks given to attention are worth $+0.060$ [$+0.022$, $+0.103$] and $+0.048$ [$+0.007$, $+0.096$], and the window cost, $W = 8$ minus full attention, is -0.056 [-0.097, -0.022]. Predictions under $W = 8$ agree with the last-chunk-only predictions on 85% of items, more than with any other condition. On this benchmark the cross-chunk state contributes nothing measurable, while the same blocks given to attention do. The state is not empty, however: a linear probe recovers task variables from it that the model's own readout does not (below).

Information present, readout absent. On a single synthetic scene (eight counterfactual videos, 16 questions per task), a model fine-tuned for 64 steps under a 16-block window answers content and time queries about events outside the window at 9/16 and 6/16, against 16/16 and 15/16 with the window removed. A linear probe fitted on the recurrent state at the end of the event window recovers the color variable 16/16 and the time variable 16/16, falling to 10/16 for time by the point at which the question begins. The information is present in the state and absent from the readout; under this budget the recipe does not learn to read it back.