COUNTROUTE: PATTERN READOUT AND EVIDENCE-GROUNDED COUNTING FOR STREAMING VIDEO COUNTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Counting in a streaming video is usually treated as a single numerical inference over an increasingly long visual context. We argue that the first decision is how the count can be established. For some events a temporal pattern — the recurrence of a visual state, or an abrupt change of shot — is enough to read the count off directly. For others the count has to be grounded in semantic evidence: occurrences must be found, told apart, and added up. Video language models treat both as one direct prediction, and do poorly where a pattern would have sufficed. Evidence-grounded counting brings a second choice. Inspecting short visual blocks keeps local evidence, but each additional empty block is another opportunity for a false positive, and an event spanning blocks may be counted twice; reading a continuous textual record avoids this block-wise accumulation, but cannot recover events it never wrote down. We introduce CountRoute, a training-free streaming agent built on a query-agnostic memory of the stream. It reads a count directly off a temporal pattern only after verifying that the corresponding signal is present; otherwise it uses the record to select relevant cached blocks, and moves from inspecting their frames to reading the record as the number of selected blocks — and hence the exposure to accumulation error — grows. On SVCBench, CountRoute achieves the best overall score, ahead of existing offline and online methods.

1 INTRODUCTION

Counting in a streaming video asks a deceptively simple question again and again: how many times has an event occurred so far? Unlike offline counting, every answer is restricted to the observed prefix and becomes one point on an evolving count trajectory (Liu et al., 2026). An error is therefore not confined to a final prediction: one missed or duplicated occurrence can persist through every query point that follows. What must be counted also varies widely — from a collision over in a few frames, to an episode lasting minutes, to the cycles of a steady motion or the cuts between shots.

These cases have been approached with different mechanisms in specialised settings. Repetition counters estimate periodicity from temporal self-similarity (Dwibedi et al., 2020; Hu et al., 2022), while long-video systems retrieve frames or reason over descriptions (Zhang et al., 2024a; Wang et al., 2024; 2025b). A general video language model is told none of this structure. It is asked to map every question and video prefix directly to a number, thereby treating all counting questions as the same computation. Giving that inference more frames does not make it reliably exact: as the count grows, the model tends to settle on a plausible numerosity rather than maintain a set of identified occurrences. Streaming video models (Chen et al., 2024; Zhang et al., 2025; Qian et al., 2025; Xu et al., 2026a; Yang et al., 2025) compress the observed prefix into a memory, but a generic memory need not preserve which observations belong to one occurrence and which similar observations belong to different ones.

We argue that the first decision in video counting is not which model should answer, but how the count can be established. For some questions an available temporal pattern is sufficient. A scene transition produces a visual discontinuity; a steady repetition produces returns of a visual state. Their count can be read directly off that pattern, without semantically telling one occurrence from the next. For the remaining questions no such pattern determines the answer. Each occurrence must instead

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.

be recognised, separated from similar observations, and counted from evidence. Video language models nevertheless treat both kinds as one direct prediction over the prefix. Where a pattern would suffice, this is a mismatch of computation, not merely a shortage of visual context.

Grounding a count in evidence has its own failure mode. The natural way to avoid holding a long visual prefix is to divide it into blocks, count each block at full resolution, and add the results. This replaces one large-count inference with a sequence of local, small-count inferences. The sum, however, accumulates error with the number of blocks it absorbs. A positive count can be triggered by one ambiguous observation, whereas zero requires ruling the event out throughout the block; every empty block is thus another opportunity for a false positive. An occurrence spanning a boundary can also be seen and counted independently on both sides. Divide-and-sum trades long-context difficulty for accumulation error.

A textual record written continuously as the stream arrives offers a different representation. Read as one passage, it does not make one independent counting decision per block, so it avoids block-wise drift and preserves the identity of an episode across visual boundaries. Its loss is irreversible: an event that was too brief or incidental to be written down cannot later be recovered from the text. Local visual blocks and a continuous record therefore expose different failure modes. The event's true granularity is not available to a system at inference time, but the number of blocks retained as evidence is. That number is an observable measure of how many local counting decisions, empty-block opportunities, and boundaries a sum must absorb.

We introduce **CountRoute**, a training-free agent that routes not among models, but among ways of establishing a count (Figure 1). As the stream arrives, and without the question, it keeps a query-agnostic memory: a vision–language model writes one caption per short window; its vision encoder passes once over every frame, and each abrupt change of picture this reveals is confirmed or rejected by the same model from the frames on either side; and frames are cached in fixed blocks. At a question's first query point, the count is treated as pattern-readable only if one of two direct readouts, for recurrences or for transitions, is verified to apply on the question and the opening of the stream. Every other question is evidence-required. The record then acts as an evidence gate, naming the cached blocks worth inspecting and excluding the empty blocks that drive positive drift. When the number of selected blocks reaches a threshold, CountRoute stops adding local counts and instead reads the count once from the record. The decision uses only information available online and directly tracks the sum's exposure to accumulation error.

On SVCBench CountRoute achieves the highest overall score among the systems evaluated, ahead of existing offline and online methods. The gains follow the failure analysis: the two readouts recover transitions and repetitions that evidence-grounded counting largely misses, selecting blocks improves the subsets with many empty blocks, and reading the record contributes most on episodic events. Our contributions are:

- a view of streaming counting in which a count is either read directly off a temporal pattern or grounded in semantic evidence, together with an analysis of how block-wise evidence accumulates error and how visual blocks and textual records lose different evidence;
- CountRoute, a training-free agent over a query-agnostic memory that reads a count off a pattern only once the readout is verified to apply, and otherwise grounds it in evidence, inspecting frames or reading the record according to the number of selected blocks;
- an evaluation that builds the system one component at a time and shows that each gain lands on the subset predicted by the corresponding failure analysis, with additional tests isolating the recurrence readout and extending the empty-block regime.

2 Related Work

Counting in video. Repetition counters recover periodic structure from the video itself: an early online counter estimates the cycle length from blocks of frames as they arrive (Levy & Wolf, 2015), later ones refine it coarse to fine (Zhang et al., 2020) or predict a period from temporal self-similarity (Dwibedi et al., 2020), TransRAC correlates features at several temporal scales (Hu et al., 2022), ESCounts matches exemplars across repetitions (Sinha et al., 2024), and CountLLM exposes periodic features to a language model (Yao et al., 2025). General vision–language models, by contrast, count unreliably even in still images (Paiss et al., 2023). Long-video counting benchmarks instead treat

counting as finding and telling apart evidence; EC-Bench scores enumeration, temporal grounding and the final count separately (Tsuchiya et al., 2026). Each line fixes in advance whether its count is read off a pattern or grounded in evidence. SVCBench (Liu et al., 2026) puts actions, transitions, episodes and repetitions behind one streaming interface, so the choice has to be made question by question; that is the choice we address.

Streaming and long-video understanding. Online video models answer as frames arrive, holding the past in a learned compressed or bounded memory (Chen et al., 2024; Qian et al., 2024; Zhang et al., 2025; Qian et al., 2025; Di et al., 2025; Xu et al., 2026a; Yang et al., 2025), as do memory-augmented long-video models (Song et al., 2024; He et al., 2024). They are built, and benchmarked (Lin et al., 2024; Niu et al., 2025), for general understanding, not for keeping distinct occurrences apart over time. Long-video systems instead read captions with a language model (Zeng et al., 2023; Zhang et al., 2024a), summarise them hierarchically (Islam et al., 2024), choose which clips to inspect (Wang et al., 2024; 2025b), or perceive short clips independently and aggregate (Pang & Wang, 2025). Such divide-and-conquer, also applied to long text (Zhang et al., 2024b), is motivated as a remedy for long context, with the error of aggregating partial results as one noise term among several (Xu et al., 2026b). For counting that term is the problem: every local count enters the sum, and vision–language models are poor at reporting that something is absent (Li et al., 2023; Lovenia et al., 2024), so each empty block is another chance to add an occurrence that never happened. Our record is written without the question, like LLoVi's captions, but it serves first to decide which blocks to inspect, and is read as the answer only once the sum has grown long enough to be unreliable.

Tools for visual reasoning. VisProg and ViperGPT compose vision modules into a program for each query (Gupta & Kembhavi, 2023; Surís et al., 2023), and ReAct interleaves reasoning with actions (Yao et al., 2023); Toolformer and HuggingGPT let a language model decide which tool or model to call (Schick et al., 2023; Shen et al., 2023). Our two readouts are fixed rather than composed, and the question is not whether one can be called but whether it should be: each is used only once its own instrument confirms that the signal it reads is present. Unlike learned shot detectors (Souček & Lokoč, 2024) or frame differencing, our transition detector differences vision-encoder embeddings and has a vision–language model confirm each candidate.

3 METHOD

3.1 PROBLEM SETTING

A video is watched as it plays. At a set of *query points* $t_1 < \cdots < t_K$ the system is asked how many times some event has happened so far. An answer at t_k may depend only on $[0, t_k)$: no frame after the query point, and no information derived from one, may enter it. Each question is answered K times over the same video, and the counts are cumulative. We also require the answer at t_k not to depend on when the question was asked before: two askers who reach t_k with different earlier query points must receive the same count. A system that cuts the video at the query points it has been given reads the asker's schedule rather than the video, and on a benchmark whose query points happen to fall one occurrence apart that schedule already does much of the counting.

3.2 QUERY-AGNOSTIC STREAMING MEMORY

Everything computed before a question arrives forms a query-agnostic memory of the stream: a *record* of captions, a set of confirmed *transitions*, and frames cached in *blocks* (Figure 1, left). None of it sees the question, so it is built once per video and shared by every question about it.

Both constraints are enforced at segmentation time rather than checked afterwards. Record windows and frame blocks lie on fixed grids of video time, w and B seconds long, that no query point interrupts; frames are cached into their block as they arrive. At t_k the agent uses the blocks

$$\mathcal{B}(t_k) = \big\{[iB, (i+1)B)\big\}_{0 \le i < m_k}, \qquad m_k = \lfloor t_k / B \rfloor, \tag{1}$$

those that have ended by then, and the windows given by the same expression with w in place of B. No frame after t_k can reach the answer given at it, a transition is used only once the frames it was detected from have been seen, and since Eq. (1) depends on t_k alone, the pieces read at t_k are the same whatever the earlier query points were.

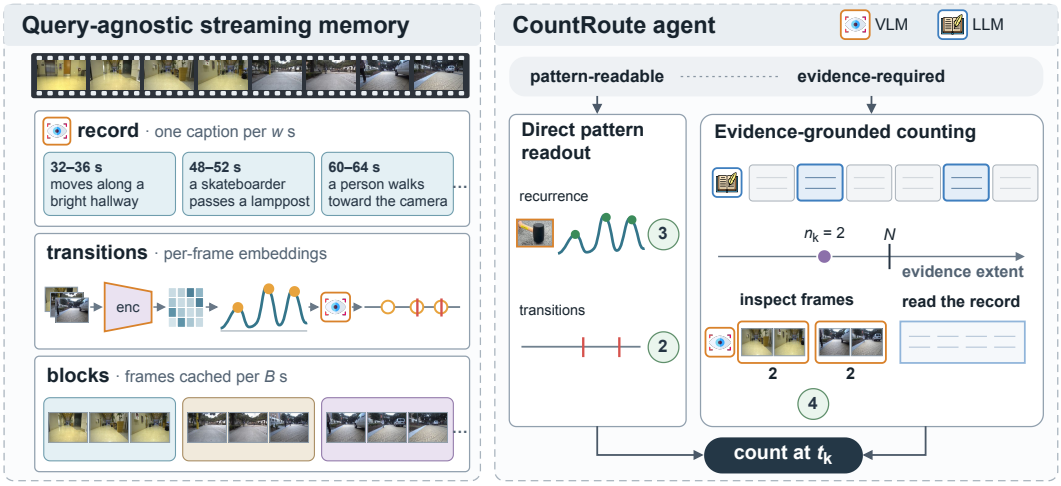


Figure 1: **CountRoute keeps a query-agnostic memory of the stream and chooses how to count at query time.** Left: without the question, a vision–language model writes a record of one caption per w-second window; per-frame embeddings yield candidate transitions (amber), which the same model confirms or rejects from the frames on either side (red); and frames are cached in fixed B-second blocks. Right: once per question, at the first query point, the agent decides whether the count is pattern-readable — recurrences of a key state (`periodic`) or confirmed transitions (`cuts`) — and if so reads it directly off that signal. Otherwise the count is grounded in evidence: a language model marks the blocks whose record supports the event; while the number of marked blocks n_k is below N, the visual model counts each block and the counts are summed, and from N on the language model reads the record over $[0, t_k)$ once. Orange: vision–language model (VLM); blue: language model (LLM).

Record. Each window of a fixed length w is described once by a vision–language model that is given no question. The model is asked for one thing only: what actions or events occur in this window, and when, stated concretely and literally against the frame timestamps. It is not asked to summarise the window, to judge what matters, or to look beyond it.

Transitions. Every frame also passes once through the vision encoder of the same model, and a transition detector runs on these embeddings. At each frame it compares the mean embedding of a short span before the frame with that of the span after it; the dissimilarity, divided by its median over the preceding stretch, peaks where the picture changes all at once. Peaks above a threshold are candidates, and within a short neighbourhood only the highest is kept. A candidate is fixed at the first query point by which the frames it depends on have been seen, and is not revised afterwards. It is then shown to a vision–language model as the frame just before it and the frame just after it, with no question. The model first lists what is visible in both frames, then says whether the picture has changed shot or only moved, choosing from a closed list of change types. A candidate it calls a change of shot is a *transition*.

3.3 The CountRoute agent

At a question's first query point the agent decides, once, how the count will be established, and keeps the choice $r(q)$ fixed for the question:

$$r(q) = \begin{cases} \texttt{periodic} & \text{if } \mathrm{VLM}_{\mathrm{steady}}\big(q,\, V_{[0,\,\min(t_1, T_0))}\big) = 1, \\ \texttt{cuts} & \text{else if } \mathrm{LLM}_{\mathrm{cuts}}(q) = 1 \text{ and } \exists\, c \in \mathcal{C} :\ c \le t_1 - \delta, \\ \texttt{evidence} & \text{otherwise.} \end{cases} \tag{2}$$

$\mathrm{VLM}_{\mathrm{steady}}$ is shown the opening stretch of the available prefix, at most T_0 seconds long, and answers, with evidence, whether the counted action recurs at a steady rate; if the stretch is too short to establish repetition, it answers no. $\mathrm{LLM}_{\mathrm{cuts}}$ reads the question alone and recognises whether it

asks about changes of shot. $\mathcal{C}$ is the set of transitions, and δ the delay after which one is usable. A question is *pattern-readable* when $r(q)$ is one of the two readouts, and *evidence-required* otherwise. This is not a general test of whether a count can be read off some pattern: the agent recognises only the two patterns it has readouts for, and only once the stream shows them. Wording alone commits a question to neither readout, and no routing confidence threshold is added; the detector's own parameters are given in §4.

3.4 DIRECT PATTERN READOUT

Some counts are better read off a temporal pattern than obtained by identifying each occurrence in the record or in cached frames. `cuts` returns the number of transitions in $[0, t_k)$ when the question concerns changes of scene. `periodic` counts how often one visual state recurs. At the first query point, a vision–language model reads the question and picks, from frames at the opening of the prefix, the one that best shows the counted action; this frame is the key state. At each query point, every frame of $[0, t_k)$ is compared with the key state after the mean embedding of the prefix has been subtracted from both — without the subtraction the frames of one clip all look alike to the encoder — and the count is the number of peaks of that similarity whose prominence exceeds a fixed fraction of its typical swing. The two readouts read different signals; neither adds per-block event counts.

3.5 EVIDENCE-GROUNDED COUNTING

Inspecting frames. For an evidence-required question, the initial reader is `inspect`. Each opened block is re-watched from its cached frames in one vision–language model call; the returned counts are added. A completed block is counted once and its count reused at every later query point, so each query point adds at most the blocks completed since the previous one. The block length trades two errors against each other: a short block is more often empty and can report an occurrence that is not there, and a long one more often holds several occurrences, which a single pass tends to count as one.

Evidence gate. The record acts as an evidence gate before a block is inspected. At each query point, a language model, the selector, is shown the blocks that have arrived since the previous point, each listed with the record's descriptions that fall inside it, and names the blocks whose descriptions give a positive reason to expect the counted event: they name the event, its associated activity, or the moment at which it would occur. The setting alone is insufficient. A named block is opened; a block left closed is never inspected and contributes zero to the sum. Thus a clearly unsupported empty block cannot add a hallucinated local count. The gate can also miss a real event that the record failed to describe; selection trades that risk against the repeated false positives of counting every empty block.

Frozen verdicts. Each block is judged once, when it first arrives, and the verdict is never revised, so the selector reads only newly arrived descriptions at each query point. Its decision concerns where to look, not how many events occurred. Writing $\mathcal{S}_j$ for the blocks named at t_j, the blocks opened at t_k are

$$\mathcal{O}_k = \begin{cases} \mathcal{B}(t_k) & \text{if } |\mathcal{B}(t_k)| < G, \\ \left\{\, b \in \mathcal{B}(t_k) :\ b \cap s \neq \emptyset \text{ for some } s \in \bigcup_{j \le k} \mathcal{S}_j \,\right\} & \text{otherwise.} \end{cases} \tag{3}$$

A verdict given at a later query point never enters an earlier answer, and a block any part of which has been named stays open. While the prefix holds fewer than G blocks every block is counted, whatever the selector said about it: a record that short gives too little to contrast one block against another, and we would rather not drop a block on that evidence.

Reading the record. Selection cannot remove a block that an event crosses into, and such an event can be counted in both blocks; these duplicates accumulate as more selected blocks enter the sum. The other reader, `read`, reads the record over $[0, t_k)$ as one passage and returns one count: it can miss events the record omits, but adds no local counts across boundaries.

The rule. With $n_k = |\mathcal{O}_k|$ the number of blocks the sum would add, the answer at t_k is

$$\hat{y}_k = \begin{cases} \mathrm{Per}(t_k) & \text{if } r(q) = \texttt{periodic}, \\ \big|\{c \in \mathcal{C} : \, c \le t_k - \delta\}\big| & \text{if } r(q) = \texttt{cuts}, \\ \sum_{b \in \mathcal{O}_k} \mathrm{cnt}(b) & \text{if } r(q) = \texttt{evidence} \text{ and } n_k < N \quad (\texttt{inspect}), \\ \mathrm{Read}\big(q,\, \mathcal{R}_{<t_k}\big) & \text{if } r(q) = \texttt{evidence} \text{ and } n_k \ge N \quad (\texttt{read}), \end{cases} \tag{4}$$

where $\mathrm{Per}(t_k)$ is the recurrence count of `periodic`, $\mathrm{cnt}(b)$ the count returned by inspecting block b, computed once and reused, and $\mathcal{R}_{<t_k}$ the record over $[0, t_k)$. The switch between the last two cases calls no model. n_k is a proxy for how much error the sum has had the chance to accumulate — every further block is another chance for a false occurrence and another boundary to count one twice — not a measurement of how long one occurrence lasts, which the agent cannot observe.

4 Experiments

4.1 Benchmark and setting

We evaluate on the event-counting part of SVCBench (Liu et al., 2026), a streaming video counting benchmark; this part has 454 questions over 281 videos, asked at 2279 query points in total. Videos run 181 s at the median and up to 2273 s. The benchmark groups these questions into four subsets: **E1-Action** counts a named action performed by a person or a vehicle; **E1-Transit** counts changes of what the picture shows; **E2-Episode** counts stretches of time during which something named is happening; and **E2-Periodic** counts repetitions of one short action. Each question is asked at several query points over the same video — one to five for most, and thirteen to fifteen for fifty questions in E1-Action — and the answers are cumulative.

Two further sets appear in the ablations, each for something SVCBench cannot show.

PushupBench (Li et al., 2026) is 446 clips of exercise footage cut from 34 full-length videos by 11 creators, asking how many repetitions of a named movement a clip contains. We use the 227-clip split released publicly — the same data the authors' evaluation task loads — and report the metrics that task reports: exact match, mean absolute error, R^2, and off-by-one accuracy. We need it because SVCBench's periodic subset is built by repeating one short clip end to end: its file names carry the repetition count, which matches the annotated event count exactly; the intervals between consecutive events repeat as a fixed pattern, far more regular than on the other three subsets; and each video ends on its last event. Every repetition there looks the same, which is what `periodic` relies on when it counts the returns of one state, so that subset can only establish a necessary condition. One occurrence is a single repetition and a clip carries one query at its end, so the set measures how accurately the repetitions are counted, not how a count is held along a stream.

Goal-Step sparse is 31 questions over 30 Ego4D videos (Grauman et al., 2022), built from the Goal-Step annotation (Song et al., 2023) by keeping step types that can be counted independently of one another: at least three occurrences, at least five seconds between them, a median single occurrence under two minutes, and no umbrella step names. We need it because block selection can only remove blocks that hold nothing, so the room it has to work in is bounded by how many of them there are. Over the 40 s blocks we use, the fraction holding no occurrence is 0%, 40%, 58% and 63% on E2-Periodic, E1-Transit, E1-Action and E2-Episode; on this set it is 78%, which extends the range of that variable instead of repeating it.

4.2 Metrics

We report the three metrics the benchmark defines, as defined in its paper, and aggregate them as it does: each metric is computed within a question first, then averaged over questions, so that a question asked at five points does not outweigh one asked at three.

GPA scores a count p against ground truth g by $\exp\big(-(p-g)^2/2\sigma^2\big)$ with $\sigma = 0.05\max(g, 1)$. The tolerance is relative, so at small counts it is effectively exact match: $g = 2$ admits ± 0.1.

MoC measures monotonicity. Counts are cumulative and may only grow; the metric reports the fraction of the sequence traversed before the first decrease.

UDA measures updating rather than correctness: over the steps at which the true count changes, the fraction at which the prediction also moved. Steps at which the truth does not change are excluded, and a question whose truth never changes has no UDA.[1]

4.3 IMPLEMENTATION

The record is written by Qwen3-VL-8B (Bai et al., 2025a), one caption per window of $w = 4$ s, at 2 fps scaled to 448 pixels wide, at most 500 tokens per window; the only other thing run on the stream is its vision encoder, once per frame, for the transition detector and for `periodic`. Windows lie on a 4 s grid of video time and are written once per video. A query point that does not fall on a grid line also uses the stretch since the last one, as a shorter window and a shorter block used at that point only; by the next query point the complete window or block has taken its place. The decision agent uses MiniMax-M3 (MiniMax, 2026) for every text decision — deciding whether `cuts` applies, reading the record, and naming which blocks to inspect — and Qwen3-VL-8B for inspecting blocks, for confirming transitions, for picking the key state, and for the check that guards `periodic`. Blocks are $B = 40$ s long. The granularity decision calls no model: the count is read off the record once at least $N = 3$ selected blocks would have to be summed, and the selector's verdicts are acted on once the prefix holds at least $G = 4$ blocks. Inspecting samples 32 frames from a block. A shorter block is judged at its own query point and again once it is complete, and one under 0.5 s is not counted. The selector judges at most 16 blocks in one call. All eight prompts are released verbatim in the supplementary material.

Transitions are detected on Qwen3-VL-8B's own vision encoder: frames at 12 fps, 448×448, mean-pooled over patches and normalised. The dissimilarity at a frame compares the mean of the six frames before it with the six from it onwards, half a second on each side, and is divided by its median over the preceding 10 s. Peaks of at least 1.0 are candidates, and within ± 1 s only the highest is kept. Each candidate is confirmed from the frames 0.5 s before and after it, and a candidate is used from the first query point at least $\delta = 0.58$ s after it — the half-second the detector looks ahead, and one frame. `periodic` picks its key state from 16 frames spread over the opening of the prefix, up to 8 s and ending at the first query point, and counts a peak once its prominence reaches half the median range of the similarity over ± 5 s windows, with peaks at least 0.4 s apart. The check that guards `periodic` shows the model the opening of the prefix at 1.5 fps — up to $T_0 = 40$ s, and less when the first query point comes sooner, so 28 s and 42 frames at the median — and asks for a yes or no. The rate is low because portrait video costs three times the tokens per frame.

4.4 MAIN RESULTS

Table 1 places our system in the benchmark's own leaderboard. The closest comparison in the table is Qwen3-VL-8B, the vision–language model we ourselves call, run offline on 64 frames of the prefix at each query point.

Reading a pattern where one suffices. CountRoute has the highest GPA overall and on three of the four subsets, 38.4 against 32.0 for the best offline method and 18.7 for the best streaming system. The largest gaps are on E2-Periodic and E1-Transit, where a readout applies.

Grounding the count in evidence where none does. On E1-Action and E2-Episode no readout applies, the margin over offline models is smallest, and the gain rests on selection and on reading the record. On E1-Action the margin over Gemini-3-Flash is narrow (29.2 against 28.5), and three proprietary models are ahead on E2-Episode, Gemini-3-Flash by 4.2 points (41.7 against 37.5).

4.5 ABLATIONS

What each component is worth. Table 2 builds the system up one component at a time, starting from the barest evidence-grounded policy: sum the counts of every completed block. Each row adds

[1] We follow the definition in the benchmark paper. The reference implementation released with the benchmark computes a different quantity — it compares the sign of the change over *every* adjacent pair, counting a step where neither truth nor prediction moves as correct — which inflates the score on subsets where the truth is mostly static. All UDA numbers we report, for our system and for the baselines we measured, follow the paper.

Table 1: **Main results.** Each cell is GPA / MoC / UDA. The groups down to *Open-source, online* are the benchmark's own and the numbers are as reported in its paper; CountRoute is measured on the released annotation at the same query points. Bold is the best value in a column, Human aside. Overall is weighted by question count $(203/48/147/56)$. UDA follows the definition in the benchmark's paper, Eq. (3): over the steps at which the true count changes, the fraction at which the prediction also moved. The code released with the benchmark computes a different quantity — the direction of change over *every* adjacent pair — so the UDA column is best read within a row rather than between reported and measured rows.

Method	E1-Action	E1-Transit	E2-Episode	E2-Periodic	Overall
Human					
Human	94.9 / 100.0 / 99.3	98.3 / 100.0 / 100.0	97.0 / 100.0 / 99.3	93.2 / 100.0 / 100.0	95.7 / 100.0 / 99.5
Blind LLM					
GPT-4-Turbo	13.1 / **96.5** / 4.2	23.6 / 95.8 / 4.2	21.9 / **93.7** / 7.1	0.0 / 94.2 / 5.4	15.4 / **95.2** / 5.3
Proprietary					
Gemini-3-Flash	28.5 / 66.0 / 61.8	49.8 / 83.7 / **87.3**	**41.7** / 79.5 / 78.0	3.9 / 56.2 / **88.8**	32.0 / 71.0 / 73.1
Doubao-Seed-1.8	24.0 / 76.9 / **70.7**	38.2 / **96.4** / 82.1	40.5 / 84.4 / 79.3	0.8 / 50.4 / 81.7	28.0 / 78.1 / **76.0**
Kimi-K2.5 (Kimi Team, 2026)	21.9 / 66.0 / 70.3	33.5 / 74.8 / 83.9	38.7 / 78.7 / 74.3	0.4 / 52.2 / 63.4	25.9 / 69.3 / 72.2
GPT-5.4	24.2 / 71.5 / 58.5	42.0 / 85.9 / 82.1	37.5 / 79.1 / 70.6	3.2 / 40.2 / 45.1	27.8 / 71.6 / 63.3
Open-source, offline					
Qwen3-VL-8B (Bai et al., 2025a)	22.5 / 84.9 / 54.5	36.9 / 95.3 / 70.8	35.6 / 82.1 / 65.4	0.0 / 68.8 / 29.0	25.5 / 83.1 / 56.6
Qwen3-VL-30B (Bai et al., 2025a)	22.5 / 88.1 / 46.4	33.6 / 92.7 / 75.9	35.3 / 89.0 / 64.5	2.5 / 68.8 / 37.5	25.4 / 86.5 / 54.3
Qwen2.5-VL-7B (Bai et al., 2025b)	9.5 / 66.7 / 33.3	13.3 / 90.6 / 23.4	11.0 / 66.5 / 37.9	0.0 / 65.2 / 32.1	9.2 / 69.0 / 33.6
InternVL-3.5-8B (Wang et al., 2025a)	22.5 / 80.2 / 52.1	20.0 / 86.8 / 77.1	31.8 / 81.3 / 65.3	1.0 / 66.5 / 30.4	22.6 / 79.6 / 56.3
Molmo2-8B (Clark et al., 2026)	3.5 / 71.7 / 21.7	2.7 / 95.5 / 7.6	9.4 / 73.7 / 35.3	0.4 / 66.1 / 30.4	4.9 / 74.2 / 25.7
Qwen3.5-35B-A3B	19.0 / 78.1 / 47.6	27.7 / 87.2 / 80.6	29.3 / 83.9 / 67.1	0.8 / 73.1 / 26.9	21.0 / 80.3 / 54.8
Open-source, online					
StreamingVLM (Xu et al., 2026a)	15.8 / 73.4 / 40.8	13.8 / 58.5 / 61.3	20.7 / 63.3 / 54.9	0.4 / 82.6 / 24.1	15.3 / 69.7 / 45.5
Dispider (Qian et al., 2025)	11.8 / 79.8 / 17.0	8.9 / 67.5 / 15.8	7.6 / 84.0 / 14.7	0.0 / 96.4 / 4.9	8.7 / 81.9 / 14.6
LiveStar (Yang et al., 2025)	16.1 / 89.5 / 23.5	22.8 / 91.7 / 31.9	23.7 / 87.1 / 38.8	11.1 / 93.2 / 8.9	18.7 / 89.4 / 27.5
Flash-VStream-7B (Zhang et al., 2025)	19.0 / 75.5 / 31.0	11.9 / 84.2 / 41.8	17.2 / 77.0 / 35.7	0.0 / 76.4 / 19.9	15.3 / 77.0 / 32.3
CountRoute	**29.2** / 83.8 / 56.6	**58.9** / **96.4** / 86.3	37.5 / 87.5 / **79.4**	**56.4** / **97.3** / 85.7	**38.4** / 88.0 / 70.7

Table 2: **Building the system up** (GPA). Each row adds one component to the row above and keeps it, so the last row is the full system. *Direct* is the reference point: 64 uniformly sampled frames of $[0, t_k)$ in one call, with no record and no blocks.

	E1-Action	E1-Transit	E2-Episode	E2-Periodic	Overall	Δ
Direct, 64 *frames*	23.7	41.5	31.3	1.7	25.3	
Sum every block	22.5	28.2	21.6	1.2	20.2	
+ block selection	26.6	27.9	27.5	1.2	23.9	+3.7
+ read the record	**29.2**	28.1	**37.7**	4.0	28.7	+4.8
+ `cuts`	**29.2**	**58.9**	37.5	4.0	31.9	+3.2
+ `periodic` (full)	**29.2**	**58.9**	37.5	**56.4**	**38.4**	+6.5

one part and keeps it. We add them in dependency order, because the granularity rule counts *selected* blocks and so is only defined once selection exists.

Every step is positive, and each one lands on the subset it was meant for. Summing every completed block is the weakest policy in the table: at 20.2 it is below answering directly from 64 frames, because a block that holds nothing still returns a count now and then, and a long prefix holds many of them. Selection is what makes the sum usable, and reading the record pays most on E2-Episode, whose events are long and cross block boundaries. The two readouts then each recover one subset:

Table 3: **Direct readout versus evidence-grounded counting on the same query points** (GPA). Each row is the set of points routing sent to that readout; the last two columns are what the two evidence readers return on exactly those points.

Readout	Subset	Q	Points	The readout	`read`	`inspect`
`cuts`	E1-Transit	33	155	**70.6**	26.6	25.2
`periodic`	E2-Periodic	46	230	**68.7**	7.0	1.4

Table 4: **Block selection against the empty-block fraction** (GPA). All rows run the `inspect` reader on every query point, so the comparison is the reader against itself and not the routed system. *Empty* is the share of 40 s blocks in which the annotation places no countable event; *Sel.* is the share of completed blocks the selector keeps by the last query point. $*$ marks a gain whose 95% bootstrap interval over questions excludes zero.

Subset	Q	Empty	No sel.	+ sel.	Δ	Sel.
E2-Periodic	56	0%	1.18	1.18	0.00	64%
E1-Transit	48	40%	28.22	27.94	-0.28	55%
E1-Action	203	58%	22.46	26.60	$+4.14^*$	39%
E2-Episode	147	63%	21.55	27.47	$+5.92^*$	50%
Goal-Step sparse	31	78%	12.50	24.76	$+12.25^*$	32%

`cuts` takes E1-Transit from 28.1 to 58.9, and `periodic` takes E2-Periodic from 4.0 to 56.4. Part of the first is owed to the readout being able to decline: using `cuts` wherever it is nominated, without asking whether a transition has been confirmed, admits more E1-Transit questions and gives back part of its gain there.

The two readouts. Table 3 takes the query points routed to each readout and replaces it with the two evidence readers, leaving everything else fixed.

Both readouts decisively beat evidence-grounded counting on the same points: `periodic` scores 68.7 versus 7.0, and `cuts` scores 70.6 versus 26.6. On real exercise footage (PushupBench), `periodic` nearly triples the exact-match rate of answering directly (Appendix A).

Which blocks to inspect. Selection only applies to the `inspect` reader, so Table 4 sends every query point to that reader regardless of what the router would have chosen and compares the reader against itself. The one variable is whether the prefix is summed over all completed blocks or only over the selected ones.

The gain tracks the empty-block fraction: selection changes nothing when no block is empty, is within noise at 40%, and is significant once most blocks are empty, doubling GPA on Goal-Step sparse. The retained share falls correspondingly, showing that selection is an evidence gate rather than a fixed budget.

5 Conclusion

Counting in a streaming video is not one computation. A count that an available temporal pattern determines is better read off that pattern than grounded in evidence, once the readout's own instrument confirms the pattern is there. For the rest, summing per-block counts accumulates errors that a question-free record avoids, at the cost of what the record never wrote down; CountRoute uses the record to choose which cached blocks to inspect and reads the record instead as the selection grows, reaching 38.4 GPA on SVCBench without training. Checking that a readout's signal is the one the question means is left to future work.

AI USE STATEMENT

We used generative AI tools to polish the language of this paper, to search for and summarise related literature, and to assist with writing the implementation and analysis code. We have reviewed all AI-assisted work and take responsibility for the final content of this paper.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, et al. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025a.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, et al. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025b.

Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. VideoLLM-online: Online video large language model for streaming video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Christopher Clark, Jieyu Zhang, Zixian Ma, Jae Sung Park, Mohammadreza Salehi, Rohun Tripathi, et al. Molmo2: Open weights and data for vision-language models with video understanding and grounding. *arXiv preprint arXiv:2601.10611*, 2026.

Shangzhe Di, Zhelun Yu, Guanghao Zhang, Haoyuan Li, Tao Zhong, Hao Cheng, Bolin Li, Wanggui He, Fangxun Shu, and Hao Jiang. Streaming video question-answering with in-context video KV-cache retrieval. In *International Conference on Learning Representations (ICLR)*, 2025.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Counting out time: Class agnostic video repetition counting in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Kristen Grauman, Andrew Westbury, Eugene Byrne, et al. Ego4D: Around the world in 3,000 hours of egocentric video. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14953–14962, 2023.

Bo He, Hengduo Li, Young Kyun Jang, Menglin Jia, Xuefei Cao, Ashish Shah, Abhinav Shrivastava, and Ser-Nam Lim. MA-LMM: Memory-augmented large multimodal model for long-term video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Huazhang Hu, Sixun Dong, Yiqun Zhao, Dongze Lian, Zhengxin Li, and Shenghua Gao. TransRAC: Encoding multi-scale temporal correlation with transformers for repetitive action counting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Md Mohaiminul Islam, Ngan Ho, Xitong Yang, Tushar Nagarajan, Lorenzo Torresani, and Gedas Bertasius. Video ReCap: Recursive captioning of hour-long videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Kimi Team. Kimi K2.5: Visual agentic intelligence. *arXiv preprint arXiv:2602.02276*, 2026.

Ofir Levy and Lior Wolf. Live repetition counting. In *IEEE International Conference on Computer Vision (ICCV)*, pp. 3020–3028, 2015.

Shengzhi Li, Jiarun Chen, Karun Sharma, Jiaqi Su, and Shichao Pei. PushupBench: Your VLM is not good at counting pushups. *arXiv preprint arXiv:2604.23407*, 2026.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 292–305, 2023.

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

Pengyiang Liu, Zhongyue Shi, Hongye Hao, Qi Fu, Xueting Bi, Siwei Zhang, Xiaoyang Hu, Zitian Wang, Linjiang Huang, and Si Liu. SVCBench: A streaming video counting benchmark for spatial-temporal state maintenance. In *European Conference on Computer Vision (ECCV)*, 2026. arXiv:2603.12703.

Holy Lovenia, Wenliang Dai, Samuel Cahyawijaya, Ziwei Ji, and Pascale Fung. Negative object presence evaluation (NOPE) to measure object hallucination in vision–language models. In *Workshop on Advances in Language and Vision Research (ALVR), ACL*, 2024.

MiniMax. MiniMax-M3. `https://www.minimax.io/blog/minimax-m3`, 2026.

Junbo Niu, Yifei Li, Ziyang Miao, Chunjiang Ge, Yuanhang Zhou, Qihao He, Xiaoyi Dong, Haodong Duan, Shuangrui Ding, Rui Qian, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, and Jiaqi Wang. OVO-Bench: How far is your video-LLMs from real-world online video understanding? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Roni Paiss, Ariel Ephrat, Omer Tov, Shiran Zada, Inbar Mosseri, Michal Irani, and Tali Dekel. Teaching CLIP to count to ten. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Ziqi Pang and Yu-Xiong Wang. MR. Video: MapReduce as an effective principle for long video understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. Streaming long video understanding with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Rui Qian, Shuangrui Ding, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Yuhang Cao, Dahua Lin, and Jiaqi Wang. Dispider: Enabling video LLMs with active real-time interaction via disentangled perception, decision, and reaction. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Saptarshi Sinha, Alexandros Stergiou, and Dima Damen. Every shot counts: Using exemplars for repetition counting in videos. In *Asian Conference on Computer Vision (ACCV)*, 2024.

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 *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Yale Song, Eugene Byrne, Tushar Nagarajan, Huiyu Wang, Miguel Martin, and Lorenzo Torresani. Ego4D Goal-Step: Toward hierarchical understanding of procedural activities. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2023.

Tomáš Souček and Jakub Lokoč. TransNet V2: An effective deep network architecture for fast shot transition detection. In *ACM International Conference on Multimedia (ACM MM)*, 2024.

Dídac Surís, Sachit Menon, and Carl Vondrick. ViperGPT: Visual inference via python execution for reasoning. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Fumihiko Tsuchiya, Taiki Miyanishi, Shunsuke Yasuki, Mahiro Ukai, Nakamasa Inoue, Shuhei Kurita, Yusuke Iwasawa, and Yutaka Matsuo. Diagnosing long-video quantitative reasoning in multimodal LLMs via enumeration and counting. *arXiv preprint arXiv:2603.29943*, 2026.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, et al. InternVL3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *arXiv preprint arXiv:2508.18265*, 2025a.

Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. VideoAgent: Long-form video understanding with large language model as agent. In *European Conference on Computer Vision (ECCV)*, 2024.

Ziyang Wang, Shoubin Yu, Elias Stengel-Eskin, Jaehong Yoon, Feng Cheng, Gedas Bertasius, and Mohit Bansal. VideoTree: Adaptive tree-based video representation for LLM reasoning on long videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025b.

Ruyi Xu, Guangxuan Xiao, Yukang Chen, Liuning He, Yao Lu, and Song Han. StreamingVLM: Real-time understanding for infinite video streams. In *International Conference on Learning Representations (ICLR)*, 2026a.

Zhen Xu, Shang Zhu, Jue Wang, Junlin Wang, Ben Athiwaratkun, Chi Wang, James Zou, and Ce Zhang. When does divide and conquer work for long context LLM? a noise decomposition framework. In *International Conference on Learning Representations (ICLR)*, 2026b.

Zhenyu Yang, Kairui Zhang, Yuhang Hu, Bing Wang, Shengsheng Qian, Bin Wen, Fan Yang, Tingting Gao, Weiming Dong, and Changsheng Xu. LiveStar: Live streaming assistant for real-world online video understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

Ziyu Yao, Xuxin Cheng, Zhiqi Huang, and Lei Li. CountLLM: Towards generalizable repetitive action counting via large language model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Andy Zeng, Maria Attarian, Brian Ichter, Krzysztof Choromanski, Adrian Wong, Stefan Welker, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, and Pete Florence. Socratic models: Composing zero-shot multimodal reasoning with language. In *International Conference on Learning Representations (ICLR)*, 2023.

Ce Zhang, Taixi Lu, Md Mohaiminul Islam, Ziyang Wang, Shoubin Yu, Mohit Bansal, and Gedas Bertasius. A simple LLM framework for long-range video question-answering. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024a.

Haoji Zhang, Yiqin Wang, Yansong Tang, Yong Liu, Jiashi Feng, and Xiaojie Jin. Flash-VStream: Efficient real-time understanding for long video streams. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Huaidong Zhang, Xuemiao Xu, Guoqiang Han, and Shengfeng He. Context-aware and scale-insensitive temporal repetition counting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö. Arık. Chain of agents: Large language models collaborating on long-context tasks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

A DIRECT PATTERN READOUT

The transition detector. Table 5 varies the threshold on the detector's score and removes the confirmation step, scoring the 40 questions that routing nominates for `cuts` (39 in E1-Transit, one in E2-Episode). The set is fixed, so a question that stops passing the check is scored on the evidence path instead. Thresholds from 1 to 3 give the same score within two points; higher ones start to miss transitions, and at 10 seven questions no longer pass the check. Without confirmation the score falls to 26.9 at our threshold, no better than never using the readout (28.1), and even the best threshold recovers only 58.5: the confirmation step, not the threshold, is what makes the count usable.

Table 5: **Transition detector** (GPA over the 40 questions nominated for `cuts`). *Pass* is the number that clear the check; the rest are scored on the evidence path. Never using `cuts` scores 28.1.

Score threshold	1	1.5	2	3	5	10
Confirmed (ours)	64.5	65.2	**65.3**	63.4	58.8	45.6
pass	34	34	34	34	34	27
Unconfirmed	26.9	38.8	48.9	58.5	57.0	46.5

The `periodic` readout. Table 6 varies the parameters of the recurrence count and of the key-state choice, on E2-Periodic and on PushupBench. E2-Periodic is small and its videos repeat one clip, so differences of a few points on it carry little weight, and PushupBench is the set to read. The prominence fraction ρ is the one parameter that matters: 0.4 and 0.5 tie on exact match and 0.5 has the lower error; 0.3 is slightly ahead on exact match but its error is larger by a third and its off-by-one rate lower, and from 0.6 on the readout misses returns. The minimum peak spacing, the width of the window the swing is measured over, the length of the opening the key state is picked from and the number of frames it is picked among change little. What does matter is that the key state is picked with the question: every setting in which the model reads the question picks a better state on PushupBench than the frame with the least motion (22.9% exact) or a random one (16 to 22%), while on E2-Periodic, where every frame recurs, the choice barely matters.

Table 6: **The `periodic` readout** on E2-Periodic (56 questions, GPA, every question routed to the readout) and PushupBench (227 clips, as in Table 7). The first row is our setting; each later row changes one parameter.

	E2-Periodic	PushupBench		
Setting	GPA	Exact	±1	MAE
Ours: $\rho = 0.5$, spacing 0.4 s, window ±5 s, opening 8 s, 16 frames	67.2	29.1%	52.9%	4.03
$\rho = 0.3$	61.4	**30.8%**	45.8%	5.41
$\rho = 0.4$	**75.5**	29.1%	52.0%	4.52
$\rho = 0.6$	62.6	22.9%	41.9%	5.11
$\rho = 0.7$	46.6	14.1%	28.2%	6.37
spacing 0.2 / 0.8 s	67.2 / 68.3	29.1 / 27.3%	52.9 / 52.0%	4.15 / 4.14
window ±3 / ±8 s	73.0 / 66.5	29.1 / 24.2%	**53.7** / 44.9%	**4.01** / 4.38
opening 4 / 16 s (16 frames)	67.8 / 67.9	24.2 / 27.3%	46.7 / 45.8%	4.42 / 4.26
8 / 32 frames (8 s)	72.4 / 69.3	24.2 / 28.6%	45.4 / 48.0%	4.62 / 4.61

The `periodic` readout on real repetitions. SVCBench's periodic subset is built by repeating one short clip end to end, so every return of the key state there looks the same, and it can establish only a necessary condition for a readout that counts those returns. PushupBench is footage of people exercising, and asks how many repetitions of a named movement a clip contains: one query at the end of each clip, and a cycle that is real rather than constructed. Every question on it is of the kind `periodic` is for, so we send the whole set to that readout and measure the readout itself rather than the check that guards it. Table 7 reports it in the metrics the benchmark's own evaluation task reports. The baseline is the same visual model shown 64 frames of the clip and asked for the count in one call; it is not one of our readers, and stands only to give the numbers a scale.

Table 7: **The `periodic` readout on PushupBench.** The 227-clip public split, one query per clip, every clip routed to `periodic`. As in the benchmark's own evaluation, Exact and ± 1 accept any of a clip's listed counts, and MAE leaves out the clips whose error exceeds 50 (8 for each row); R^2 is computed against predicting the mean count. The benchmark paper reports results over all 446 clips, which are not comparable with these and are not shown.

	n	Exact	± 1	MAE	R^2
Direct, 64 frames	227	10.1%	23.8%	6.84	0.018
`periodic`	227	**29.1%**	**52.9%**	**4.03**	**0.403**

Table 8: **Prefix-length gate** (GPA). The selector's verdicts are acted on only once the prefix holds at least G completed blocks; $G = 0$ acts on them from the first. Same setting as Table 4, except the last row: the full routed system ($N = 3$), SVCBench overall.

Subset	G=0	G=2	G=3	G=4	G=6	G=8	G=11
E2-Periodic	1.18	1.18	1.18	1.18	1.18	1.18	1.18
E1-Transit	22.56	27.24	27.24	27.94	27.94	**28.98**	28.29
E1-Action	23.50	26.03	**27.36**	26.60	25.94	24.39	24.02
E2-Episode	26.42	26.68	27.30	**27.47**	24.53	23.62	22.61
Goal-Step sparse	**26.37**	**26.37**	25.29	24.76	21.53	20.46	19.37
Routed, overall	35.98	37.33	38.05	**38.39**	37.98	37.83	37.78

The readout nearly triples the exact-match rate of the same backbone, cuts its mean absolute error by two-fifths and raises R^2 from near zero to 0.403: counting the returns of a key state recovers something about the count that 64 frames of the clip do not. It is still far from solved — fewer than one clip in three is exactly right — and the margin is smaller than on SVCBench's periodic subset, which is what a constructed repetition against a real one should look like.

B BLOCK SELECTION

The prefix-length gate. The gate is what makes selection safe rather than merely cheap (Table 8) on SVCBench. Removing it ($G = 0$) costs most on E1-Transit and least on E2-Episode: over a prefix of one or two blocks the record gives the selector nothing to contrast them against, and it passes over blocks that hold the answer. Values of G from 2 to 4 lie within about a point of one another on each SVCBench subset; past that the gate begins to disable selection outright, and E1-Action and E2-Episode fall back towards the unselected score. Goal-Step sparse runs the other way: its prefixes are long and their opening blocks are mostly empty, so a gate that counts every early block admits exactly the blocks selection would have dropped, and $G = 0$ is best there, 1.6 points above $G = 4$. The gate protects a cold start on a prefix whose first blocks hold the answer, and costs on one whose first blocks hold nothing. We use $G = 4$ throughout.

Where the sum's error comes from. Table 9 groups query points by how many blocks have completed by then. The share of those blocks in which the annotation places nothing rises as the prefix lengthens, and the sum overshoots the true count at more and more query points: from one in twenty-five to more than four in five. Selection lowers the empty share at every length past the gate, roughly halving it from nine blocks on; it slows the rise rather than stopping it, and the overshoot rate and the score follow. The first row is the one place the two shares agree: with fewer than G blocks completed the selector's verdicts are not yet acted on, so every block is counted.

The pattern is clearest on E2-Episode, whose occurrences are coarse and whose blocks empty out fastest: there the empty share runs from 29% to 85%, the sum overshoots at 17% of query points and then at 88% or more, and selection holds the empty share near a third over the first sixteen blocks and near three-fifths beyond. On E1-Action the empty share stays under a quarter over the first four blocks and the sum undershoots at more than two query points in three; what costs it there is not empty blocks but occurrences too brief for a block to report. From five blocks on the empty share on E1-Action passes two-fifths, and the sum turns to overshooting.

Table 9: **The sum's error against the number of blocks**, over the 1975 query points outside E2-Periodic that follow at least one completed block; E2-Periodic has no empty blocks and is answered by `periodic` for the most part. Blocks are 40 s, as in Table 4. *Empty* is the share of completed blocks holding no annotated occurrence, and *of selected* the same share among the blocks the sum actually adds. *Over* is the share of query points at which the sum exceeds the true count. The true count also grows with the prefix, so the GPA columns carry that as well; the empty shares and overshoot rates do not.

		Empty blocks		Sum		+ selection	
Blocks completed	Pts	all	of selected	GPA	Over	GPA	Over
1–2	1095	8%	8%	20.5	4%	20.5	4%
3–4	317	27%	20%	12.9	26%	15.4	20%
5–8	264	48%	23%	13.1	52%	22.2	25%
9–16	198	59%	26%	8.3	65%	16.1	39%
17–32	66	81%	53%	6.9	85%	11.1	61%
33+	35	83%	43%	2.9	80%	10.1	69%

What the selector keeps. Table 10 scores the selector against the annotation: a block counts as positive if an occurrence begins inside it, the same rule that gives the empty-block shares in Table 4. Precision is above the base rate on every subset, and by the widest factor where blocks are emptiest — on Goal-Step sparse one block in five holds an occurrence and nearly one selected block in two does. The selector is discriminating, not merely keeping a fixed share.

Table 10: **Selection against the annotation.** A block is positive if an occurrence begins inside it; *with an occurrence* is the complement of the empty-block share in Table 4 and is the rate precision should be read against. Blocks are counted once per question, as judged by the last query point. E2-Periodic is a degenerate row: every block there holds an occurrence, and most of those questions are answered by `periodic` without consulting the selector at all.

Subset	Blocks	With an occ.	Selected	Precision	Recall
E2-Periodic	231	100%	64%	100.0%	63.9%
E1-Transit	271	60%	55%	90.7%	84.0%
E1-Action	1944	42%	39%	80.8%	76.0%
E2-Episode	1743	37%	50%	63.5%	86.4%
Goal-Step sparse	681	22%	32%	47.5%	70.9%

The subsets trade the errors differently. E1-Transit and E1-Action keep slightly fewer blocks than hold an occurrence; E2-Episode keeps more, and has the highest recall and, outside Goal-Step sparse, the lowest precision. That asymmetry is the long occurrence again: counting a block as positive only when an occurrence *begins* in it charges the selector for keeping the later blocks of an event that is still running. Scoring E2-Episode by overlap instead — a block is positive if any annotated interval meets it — moves its precision from 63.5% to 85.8% and its recall from 86.4% to 71.3%, so most of what the first rule calls a false selection is a block the event is still passing through.

C SWITCHING FROM INSPECTING FRAMES TO READING THE RECORD

Where the switch fires. Table 11 breaks the granularity decision down by subset. Points claimed by a readout never reach it.

Switch threshold. The score is single-peaked at $N = 3$: switching earlier loses visual evidence, whereas switching later accumulates local counting errors.

Where each reader fails. Which of the two evidence readers is right depends on how much of the video one occurrence covers, and E2-Episode is the one subset that lets this be measured: it annotates a start and an end for every occurrence, so each question can be placed by the median

Table 11: **Where the switch fires.** Points claimed by a readout never reach the granularity decision. *Switched* counts questions that reach `read` at some query point.

Subset	Q	Points	`periodic`	`cuts`	Granularity	`read`	Switched
E1-Action	203	1281	0	0	1281	333 (26%)	164/203
E1-Transit	48	205	0	155	50	15 (30%)	9/ 48
E2-Episode	147	513	0	4	509	287 (56%)	123/147
E2-Periodic	56	280	230	0	50	23 (46%)	10/ 56
All	454	2279	230	159	1890	658 (35%)	306/454

Table 12: **Where to stop summing** (GPA). $N{=}\infty$ never switches. Routing is unchanged; E2-Periodic is always 56.4.

Subset	$N{=}1$	2	3	4	5	6	8	10	∞
E1-Action	24.0	26.4	**29.2**	28.9	27.9	27.2	26.9	26.8	26.6
E1-Transit	55.0	58.6	58.9	58.9	**59.2**	**59.2**	**59.2**	**59.2**	**59.2**
E2-Episode	37.3	**38.3**	37.5	35.7	34.1	33.2	30.6	28.4	27.3
Overall	35.6	37.4	**38.4**	37.7	36.7	36.1	35.2	34.4	34.0

length of the thing being counted. (E2-Periodic also carries ends, but an occurrence there is defined to run until the next one begins, so its extent is the cycle and not an independent quantity; E1-Action and E1-Transit annotate occurrence times only.) Table 13 runs each reader over every query point of those questions, grouped by that length.

Table 13: **The two evidence readers by occurrence length** (GPA, E2-Episode), grouped by the median annotated length of one occurrence. *Inspect* sums the selected blocks at every query point, *Read* reads the record at every query point, and *CountRoute* switches between them by the rule of §3. *To record* is the share of the group's query points that rule sends to the record. The one question claimed by `cuts` is excluded.

Occurrence length	Q	Pts	Inspect	Read	CountRoute	To record
< 2 s	15	62	**31.0**	25.1	27.6	40%
2–5 s	28	105	40.3	43.9	**46.4**	43%
5–15 s	52	184	31.3	36.9	**38.3**	51%
15–60 s	33	104	16.2	**33.6**	32.2	75%
> 60 s	18	54	14.4	**43.1**	41.7	75%

Summing blocks is ahead only where an occurrence is over in under two seconds; from there on reading the record is ahead, and the margin grows with the length of an occurrence: 3.6 and 5.6 points up to fifteen seconds, then 17.4 and 28.7. A long occurrence crosses more block boundaries and is counted in more of them, whereas the record is one passage and counts it once; a brief one is more often left out of a record written without the question. The short groups hold 15 and 28 questions and are too small to carry a claim on their own, and the subset-level comparison points the same way: on E1-Action, whose occurrences are the briefest in the benchmark, summing blocks leads reading the record, 26.6 against 24.0 over 203 questions.

Neither reader alone is what the system uses. In the two middle groups the routed system is ahead of both — neither reading everything nor summing everything recovers 46.4 or 38.3 — because the rule switches per query point: early in a question, over a few blocks, the sum has had little chance to go wrong, and later the record takes over. The share of points it sends to the record rises with the length of an occurrence, from 40% to 75%, without being given it. At the ends it trails the better reader: by 3.4 points below the block sum for the briefest occurrences, and by 1.4 below the record in each of the two longest groups. The threshold counts blocks left to sum, which is a measure of how much error the sum can have accumulated rather than of how long an occurrence lasts: over these questions the number of selected blocks correlates with the length of the prefix at 0.75 (Spearman) and with the length of one occurrence at 0.31. Long occurrences cross the threshold sooner here mostly because they come with longer prefixes.

D ANSWERING IN A SINGLE PASS

A single pass stops counting occurrences as the count grows. Table 14 groups every query point at which at least two occurrences have happened by the true count there, and asks how often a single pass over 64 frames is exactly right and what it answers most often. Up to three occurrences the most common answer is the true one. Past that it is not, and from eight on it stops following the frames: more than a third of those points receive the same answer whatever is shown, and in the largest group the most common answer, at more than half of its points, is a single occurrence.

Table 14: **A single pass stops counting as the count grows** (direct, 64 frames), over the 1869 query points whose true count is at least two. *Most common answer* is the value returned at the largest share of the group's points, with that share. The 16+ group is 298 points, 232 of them E2-Periodic, whose repetitions run into the dozens; over the other three subsets alone it holds 66 points and its most common answer is the same one the 8–15 group gives.

True count	Pts	Exact	Most common answer	
2–3	466	30.0%	2	(32%)
4–7	497	8.7%	3	(22%)
8–15	608	9.0%	10	(37%)
16+	298	0.3%	1	(52%)

Sampling more frames does not close the gap. The answer a single pass returns is limited by more than how many frames it is given. Table 15 sweeps that budget on the same questions and the same groups as Table 13. Six times as many frames moves the overall score by less than two points, and in the three shorter groups 192 frames is already every frame the 2 fps cache holds — one every 0.5 to 0.8 s — so what limits the answer there is not temporal resolution but being asked about the whole prefix at once: the same model, asked block by block, returns 31.0 and 40.3 where a single pass over 192 frames returns 23.8 and 35.4. The exception is the longest group, where a single pass is ahead of both of our readers; it holds 18 questions.

Table 15: **A single pass, given more frames** (GPA, E2-Episode), grouped as in Table 13. *direct-n* answers from n uniformly sampled frames of $[0, t_k)$ in one call. *Inspect* and *Read* repeat the two readers from that table.

Occurrence length	Q	Pts	direct-32	direct-64	direct-128	direct-192	Inspect	Read
$<$ 2 s	15	62	23.4	23.9	25.4	23.8	**27.5**	25.1
2–5 s	28	105	30.4	33.8	36.4	35.4	41.8	**43.9**
5–15 s	52	184	30.0	27.5	29.7	32.4	31.8	**36.9**
15–60 s	33	104	30.4	28.8	26.3	28.9	15.7	**33.6**
$>$ 60 s	18	54	**57.5**	50.9	56.0	51.0	13.9	43.1
All	146	509	32.9	31.5	33.0	33.6	27.4	**37.0**

E BLOCK LENGTH

Block length. The overall score peaks at 40 s. E1-Action, whose occurrences are brief, is best at 40 s and E2-Episode, whose occurrences are long, at 160 s; without blocks, both lose.

Block length, without the switch. Table 17 repeats the block-length sweep of Table 16 for the `inspect` reader on its own and for the full system with G and N held fixed in seconds rather than in blocks. Summed without the switch, longer blocks keep helping up to 80 s: fewer blocks are empty and fewer false counts enter the sum. With the switch the maximum moves to 40 s, because the record now takes over the long prefixes that longer blocks would have helped, and without blocks ($B{=}\infty$) the full system loses nearly seven points. Holding G and N fixed in seconds gives the same maximum. On Goal-Step sparse, whose blocks are the emptiest, the plain sum keeps improving up to 160 s and the selected sum is best at 80 s; selection gains there at every finite block length, since the selector judges each block however long it is.

Table 16: **Block length** (GPA, full system, $G = 4$, $N = 3$). E2-Periodic is 56.4 for every setting.

B	E1-Action	E1-Transit	E2-Episode	Overall
20 s	26.2	**60.3**	35.6	36.6
40 s	**29.2**	58.9	37.5	**38.4**
80 s	27.3	59.8	36.8	37.4
160 s	24.1	60.2	**38.6**	36.6
∞	17.7	58.8	32.4	31.6

Table 17: **Block length, component by component** (GPA, SVCBench overall, and Goal-Step sparse on the `inspect` reader). *Full, seconds* holds the gate at 160 s and the switch at 120 s of selected video, the values $G = 4$ and $N = 3$ take at $B = 40$ s; it is undefined without blocks.

	SVCBench overall				Goal-Step sparse	
B	Sum	+ sel.	Full	Full, seconds	Sum	+ sel.
20 s	16.7	21.2	36.6	33.2	12.4	26.5
40 s	20.2	23.9	**38.4**	**38.4**	12.5	24.8
80 s	**22.8**	**24.9**	37.4	37.3	25.2	**30.6**
160 s	22.5	23.5	36.6	34.3	**27.4**	29.6
∞	21.5	21.5	31.6	—	25.3	25.3

F Cost

Where the computation goes. The record and the transition detector both work as the stream arrives and without the question, so what they cost is not paid when a question is asked; what matters is whether they keep up with the video. We time them as a stream would run them, one window at a time — a window cannot be processed before it has been shown — with all three on one GPU, one after another (Table 18, last column). A four-second window costs 0.32 s to decode into eight frames and 1.55 s to caption. The vision encoder's pass over the same four seconds at 12 fps takes 0.38 s, decoding included. Candidate transitions arrive at 1.13 per window on average over the benchmark, and confirming one, two frames and one call, takes 1.15 s, about half of it spent seeking to and decoding those two frames again. In all a window costs 3.55 s against the four seconds it covers: the work done as the stream arrives keeps up with it, at $1.13\times$ real time, but with little to spare. Nearly half of that is confirmation, and the candidate rate is uneven: in videos that cut often, candidates arrive faster than one window's time can confirm them, and our evaluation, which treats a confirmed transition as available once its frames have been seen, assumes those confirmations are served in parallel. Decoding is measured with the same command the pipeline uses, seeking to each window and to each confirmed frame separately; a stream would decode each frame once and share it among the three, and pay less.

Table 18 accounts for the rest. At a query point the system inspects on average 20 frames, against the 64 a direct answer reads every time, and at about half of the query points it opens no frames at all: the count is read from the record, or returned by a readout, or the blocks it needs were already counted — a block is counted once and never revisited, so of the $5{,}310$ blocks completed on the evidence path only $1{,}419$ ever reach the visual model. What the system pays instead is text: 1.49 language calls per query point, which none of the baselines make. Counting every image the full model reads, in both phases, it sends $2.3\times$ as many frames as answering directly does, most of them in the record, which is written once per video and shared by every question about it; the encoder-only pass for the detector is in addition to this.

G Limitations

Routing commits a question to a readout for good, so a wrong commitment is not a small loss: the answer becomes a readout of a signal the question did not ask about. The checks keep such errors rare but not absent. One E2-Episode question that asks for scene cuts is sent to `cuts`, whose detector confirms far more changes of shot than the annotation counts; the check establishes that the readout's instrument responds, not that it responds to what the question means. In the other

Table 18: **Where the computation goes**, over the whole benchmark (454 questions, 2279 query points). *Frames* counts images sent to the visual model. The record and the transition detector run once per video as the stream arrives and without the question, the record's partial windows once per query point; everything below them happens when a question is asked. Per-question work (the vision check, `periodic`'s choice of key state) runs at the first query point only. `cuts` calls no model at query time: it counts transitions already confirmed as the stream arrived. The encoder pass runs the vision encoder alone and no language model, so it is listed apart from the calls. *s/window* is wall-clock time per four-second window of video, run one window at a time on one GPU as a stream would; the confirmation row is 1.13 candidates per window at 1.15 s each.

	Calls	Frames	s / window
As the stream arrives			
4 s record	27,569	210,644	1.87
Transition confirmation	29,468	58,936	1.30
Vision encoder only, 12 fps	—	1,217,453	0.38
total			3.55
At query time, visual			
Vision check for `periodic`	454	15,009	
`periodic` (key state)	46	736	
Block counting	1,419	45,408	
`cuts`	0	0	
subtotal	1,919	61,153	
At query time, text			
`cuts` nomination	454	—	
Block selection	2,293	—	
`read`	658	—	
subtotal	3,405	—	
Direct, 64 frames	2,279	145,856	

direction, the vision check turns away some E2-Periodic questions whose motion is irregular yet still countable by `periodic`. A check of whether a readout fits what the question means, rather than whether its signal is present, is left to future work.