RAINCHECK: DEFERRED MULTIMODAL PREFILL FOR EFFICIENT MULTI-TURN MLLM INFERENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-turn conversations with multimodal large language models (MLLMs) reuse input context, but long multimodal inputs increase time to first token (TTFT). Query-aware hard pruning reduces prefill tokens but permanently excludes information that later questions may require. We propose RAINCHECK, a framework for deferred multimodal prefill that decouples initial token selection from later information availability. An existing pruning method selects tokens for initial prefill; RAINCHECK preserves and progressively prefills unselected tokens in source order using unused compute capacity during decode. Unfinished prefill continues after answer return during available scheduling gaps. Attention visibility control and affine state mapping enable context updates after generation begins in full-attention and hybrid recurrent models. Prefill-decode fusion reduces repeated weight reads. We evaluate Qwen3-VL-8B and Qwen3.5-9B with 30% initial multimodal-token retention. Across both models and four pruning methods, RAINCHECK recovers on average 77.7% and 60.4% of the quality lost to matched hard pruning on MultiVerse and ConvBench, respectively. On MultiVerse, the SGLang implementation reduces mean first-turn TTFT by up to 24.1% relative to full prefill, with less than 0.5% increase in mean time per output token (TPOT).

1 INTRODUCTION

Multimodal large language models (MLLMs) support multi-turn conversations about the same image or video. For example, a 30-second 720p video sampled at one frame per second corresponds to about 13,500 visual tokens under the Qwen3-VL input configuration (Qwen Team, 2025). Full input prefill processes all input tokens before generation, so long multimodal sequences increase time to first token (TTFT). Token pruning lowers TTFT by reducing prefill tokens. We target latency-sensitive interactive deployments that prioritize response time over large decode batches.

Prior work has explored pruning in modality encoders, between model layers, and during generation (Zhang et al., 2024a; Chen et al., 2024; Xing et al., 2025; Tao et al., 2025; Kim et al., 2026). Selection criteria include modality saliency, token diversity, and query relevance. Query-aware scoring prioritizes tokens relevant to the current question (Zhang et al., 2025b; Dong et al., 2026), but the selected tokens may omit information useful for the ongoing answer or later questions. Hard pruning permanently excludes unselected tokens from the reusable context. Computing missing token states requires additional prefill (Fu et al., 2024).

Deferred prefill decouples initial token selection from later information availability. Unselected multimodal tokens are processed during decode, and any unfinished prefill continues after answer return. In the small-batch setting considered here, decode is often memory-bound and leaves unused compute capacity for deferred prefill (Agrawal et al., 2023; 2024). Intervals between requests provide additional opportunities to complete deferred prefill during user reading, client-side processing, and tool execution. The scheduler limits deferred tokens per decode step to control time per output token (TPOT). Subsequent turns reuse the input states after deferred prefill completes.

We propose RAINCHECK [1], a framework for deferred multimodal prefill illustrated in Figure 1. An existing token-pruning method selects a fixed fraction of multimodal tokens as important tokens.

[1] A rain check preserves admission to a rescheduled event; RAINCHECK defers tokens for later processing.

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.

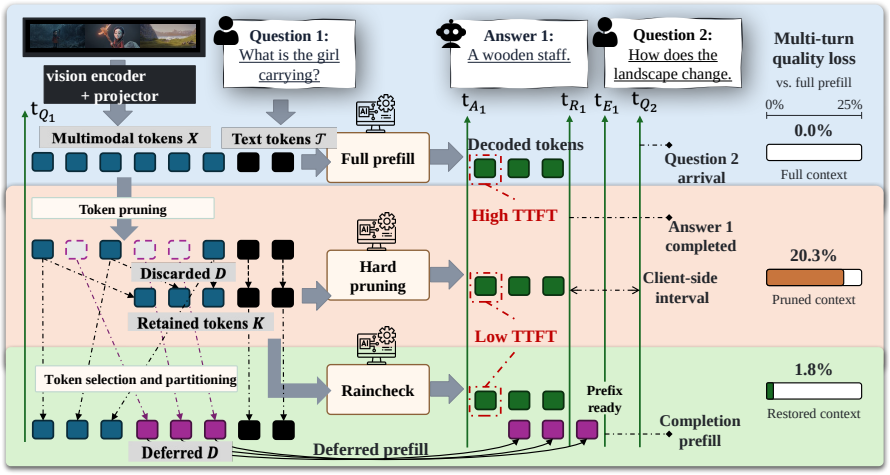


Figure 1: Schematic overview of RAINCHECK. Full prefill (`Full`) processes all input tokens before generation. Hard pruning (`Hard`) discards unselected tokens to shorten initial prefill. Raincheck (`Ours`) prefills important tokens first, processes deferred tokens during decode, and completes unfinished prefill after answer return. The next turn reuses the completed context. Blue, purple, black, and green denote important multimodal, deferred multimodal, fixed text/control, and generated tokens. Right: ConvBench mean score loss (%) relative to `Full` on Qwen3.5-9B with MMTok, 30% initial retention, and 31-token deferred chunks.

Initial prefill processes the important tokens and all text and control tokens. Input order is preserved within the important and deferred sequences. Deferred tokens are processed in consecutive chunks during decode. The co-located executor fuses prefill and decode to reduce repeated weight access. Full-attention layers use separate visibility rules for deferred and decode tokens. In hybrid models such as Qwen3.5, recurrent layers compress the processed sequence into a fixed-size state (Qwen Team, 2026; Yang et al., 2025b). Online affine state mapping updates the recurrent state using recorded transitions for trailing text and generated tokens. The corrected state supports subsequent decoding without rerunning the full sequence. To summarize, our main contributions are:

(1) *Limits of Advanced Pruning.* We identify a mismatch between one-time token pruning and multi-turn conversations: tokens omitted during prefill may become useful during generation or in later turns. Hard pruning discards them, trading reusable context for lower TTFT.

(2) *Deferred Multimodal Prefill.* We propose RAINCHECK to recover deferred tokens during decode and after answer return. Attention visibility control and affine state mapping enable context updates during generation, supported by fused prefill and decode.

(3) *Quality-Efficiency Sweet Spot.* Across two models and four pruning methods, RAINCHECK recovers on average 77.7% and 60.4% of pruning-induced quality loss on MultiVerse and ConvBench. On MultiVerse, mean first-turn TTFT falls by up to 24.1% and mean TPOT rises by less than 0.5% relative to full prefill.

2 RAINCHECK

RAINCHECK enables a model to process deferred input tokens after answer generation begins. An existing pruning method selects the initial multimodal tokens; RAINCHECK preserves the remaining tokens for later prefill. The central requirement is to extend the input context without making input states depend on generated outputs. Full-attention visibility and recurrent state mapping enforce the required dependencies, and fused execution limits repeated weight access. Figure 1 shows how deferred prefill connects the first answer to context reuse in later turns.

The main design and experiments use co-located prefill and decode. A separate prefill-decode (PD) disaggregated design places deferred prefill on a prefill worker to isolate deferred-token computation from decode; Appendix G.2 describes state transfer and the join between workers.

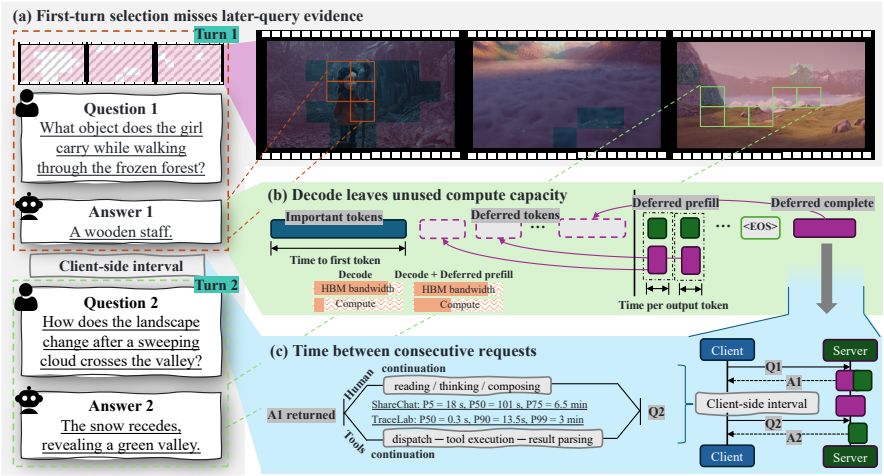


Figure 2: Motivation for deferred multimodal prefill: (a) later questions may need tokens omitted earlier; (b) unused compute capacity during decode allows deferred tokens to be processed; (c) time between turns can be used to finish deferred prefill.

2.1 Design Motivation

Initial pruning reduces prefill latency but can exclude information needed by the ongoing answer or later turns. RainCheck separates fast initial response from input recovery by scheduling deferred prefill during decode and after answer return, as illustrated in Figure 2.

Initial selection can omit useful input. Initial pruning can omit tokens useful for the ongoing answer or later turns. We compare MMTok (Dong et al., 2026) selections for the first two questions of 48 MultiVerse (Lee et al., 2025) dialogues at 30% image-token retention. In Figure 4(a), initial selection omits 19.34% of second-question selections on Qwen3-VL-8B and 17.32% on Qwen3.5-9B on average. Figure 3 illustrates a later question about an omitted jersey-number position. Adding the completed `Full` answer also changes first-question selection, as the retrospective diagnostic in Figure 4(b) shows. Initial selection should therefore decide when tokens are processed, not whether tokens remain available. Section 3.2 and Appendix D.1 evaluate restoration gains for ongoing answers and later turns.

Question 1: What color is the basketball?
Question 2: What number is on the back of the player dribbling the ball?

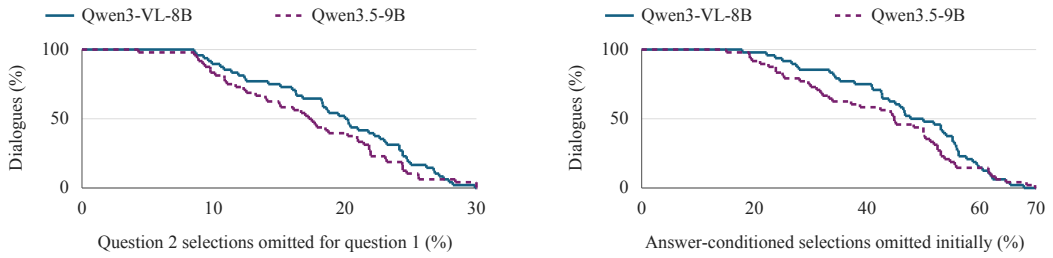


Original | First-question selection

Figure 3: A later question targets a position omitted when pruning for the first question. Purple marks the jersey number; faded cells mark omitted tokens.

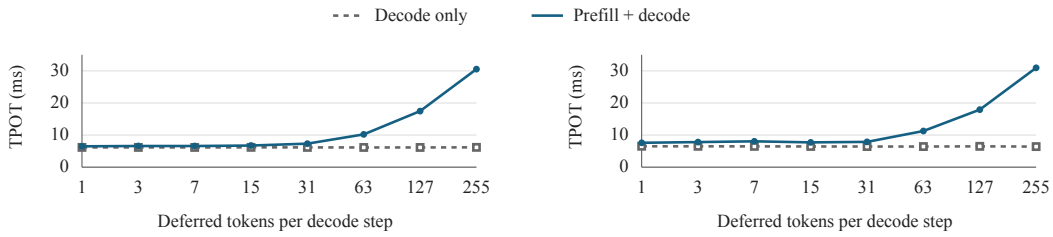
Compute capacity during decode. Deferred tokens can be processed during decode rather than delaying the first output. At small batch sizes, decode is often memory-bound and leaves unused compute capacity, as illustrated in Figure 2(b). Prefill and decode use the same model weights, so fused execution can share weight access while processing deferred tokens alongside decode (Agrawal et al., 2023; 2024). Figure 5 compares TPOT during deferred prefill with decode-only intervals at matched output positions. With 31 deferred tokens per step, TPOT increases from 6.15 to 7.33 ms on Qwen3-VL and from 6.43 to 7.88 ms on Qwen3.5. RainCheck limits deferred tokens per step to control the TPOT increase. Appendix F.2 specifies the measurement protocol.

Completion after answer return. The answer can return before deferred prefill finishes. User reading, client-side processing, and tool execution can separate consecutive requests, as illustrated in Figure 2(c). Figure 6 reports multimodal message intervals from ShareChat and individual tool-



(a) Across questions (b) After adding the first answer

Figure 4: Token-selection changes at 30% retention. Curves show the share of dialogues where first-question selection omits more than the specified percentage of (a) second-question selections or (b) question-and-answer selections.



(a) Qwen3-VL-8B (b) Qwen3.5-9B

Figure 5: TPOT by deferred chunk size versus position-matched decode-only intervals.

call durations from TraceLab (Yan et al., 2026; SyFI TraceLab, 2026). After timestamp cleaning, 72.5% of 102 ShareChat intervals last at least one minute, with a median of approximately 101 s; 61.3% of TraceLab tool calls finish within one second. Appendix F.3 defines the message intervals and tool-call durations. After answer return, RAINCHECK can finish deferred prefill if the scheduler provides enough time to process the remaining tokens before the next request.



(a) ShareChat (b) TraceLab

Figure 6: Multimodal follow-up message intervals in ShareChat and tool-call durations in TraceLab.

2.2 DEFERRED MULTIMODAL PREFILL

An existing token-pruning method selects important multimodal tokens K at the initial retention budget. RAINCHECK defers the unselected tokens D and preserves original order within both partitions to limit changes to causal attention.

Initial prefill processes K and all text, control, and special tokens, then produces the first output token. During decode, RAINCHECK prefills consecutive source-ordered chunks of D.

Each completed chunk becomes available to the ongoing answer and later turns. Generation continues with the KV cache and updated recurrent states rather than recomputing the sequence. Table 1 lists the core symbols; Appendix A.1 gives the complete notation.

Table 1: Core method notation.

Symbol	Meaning
K, D	Important / deferred tokens
$\mathcal{T}_{\text{tail}}$	Intervening input tokens
Y	Processed output tokens
S_{prefix}	Prefix state
S_{live}	Live state
P_{tail}	Tail transition matrix

States: $d_k \times d_v$. Map: $d_k \times d_k$.

2.3 Context Updates during Generation

When generation reuses the KV cache, deferred prefill must insert input states before an already processed suffix. In Figures 7 and 8, black tokens denote intervening input tokens $\mathcal{T}_{\text{tail}}$, including text, control markers, and special tokens. The sequence Y contains already processed generated tokens. Earlier context is included in the existing prefix without separate notation.

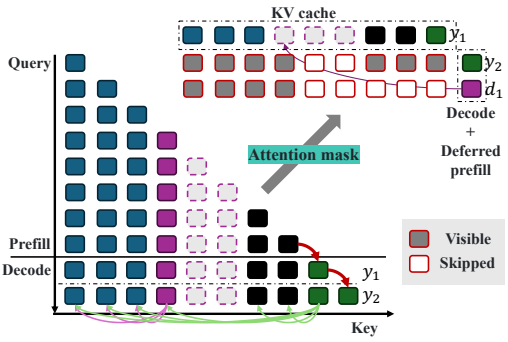


Figure 7: Deferred-prefill attention.

Full attention stores separate keys and values for each token. Gated DeltaNet (GDN) (Yang et al., 2025b) summarizes the processed sequence in a recurrent state, so extending the input requires updating the live state. Both layer types make completed chunks available to decode while keeping deferred input independent of the processed suffix.

Full-attention visibility. A deferred chunk reads the prefix processed so far and uses causal attention within the chunk, excluding $\mathcal{T}_{\text{tail}}$ and Y. The decode token reads all input processed so far and the generated history. Separate attention views enforce the two dependencies in Figure 7. New deferred KV entries precede the processed suffix; existing entries stay cached and pending deferred slots remain invisible. Appendix A.2 specifies the backend cache layouts.

GDN state mapping. For a GDN layer in Qwen3.5, S_{prefix} is the state at the recovery boundary and S_{live} is the state after $\mathcal{T}_{\text{tail}}$ and Y. Each state is a matrix in $\mathbb{R}^{d_k \times d_v}$ for one layer and head, with key dimension d_k and value dimension d_v. A state marks a sequence endpoint, not an individual token.

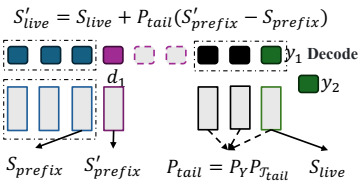


Figure 8: GDN state correction.

Recorded GDN updates compose into an affine transformation of the incoming state. The matrix $P_{\text{tail}} \in \mathbb{R}^{d_k \times d_k}$ propagates the incoming state through the suffix, and $B_{\text{tail}} \in \mathbb{R}^{d_k \times d_v}$ collects the additive token updates. The two endpoints satisfy

$$S_{\text{live}} = P_{\text{tail}} S_{\text{prefix}} + B_{\text{tail}}, \qquad P_{\text{tail}} = P_Y P_{\mathcal{T}_{\text{tail}}}. \tag{1}$$

Both factors are $d_k \times d_k$ matrices. The rightmost factor first propagates through the intervening input; P_Y then propagates through the generated tokens.

Deferred prefill advances the prefix state to S'_{prefix}. Holding the recorded suffix transitions fixed, the additive term B_{tail} cancels, giving

$$S'_{\text{live}} = S_{\text{live}} + P_{\text{tail}} \left(S'_{\text{prefix}} - S_{\text{prefix}} \right). \tag{2}$$

The prime denotes a deferred update at the same suffix endpoint. The live state already contains the additive contribution, so the correction only requires P_{tail} and the prefix-state difference. The next decode token uses the corrected live state in Figure 8.

RainCheck maintains the transition matrix as new tokens are processed. Each token applies a scalar decay and a rank-one correction to the transition matrix, requiring $O(d_k^2)$ arithmetic per head. Each deferred update propagates the prefix-state difference without rerunning the full suffix. State correction reuses recorded suffix factors and approximates a full model rerun under the expanded input. Appendix B derives the mapping and describes the short-convolution boundary repair required by Qwen3.5.

2.4 Execution and Completion

The co-located executor combines a deferred chunk and the current decode token in one pass through the decoder. Dense projections and the feed-forward network process the combined sequence, sharing weight access as in prefill-decode fusion (Agrawal et al., 2023; 2024). Full-attention layers use

separate deferred and decode views. GDN layers first advance the input state, correct the live state, and then process the decode token.

The SGLang (Zheng et al., 2024) implementation reuses fixed-shape CUDA graphs for fused execution. Common shapes combine 7, 15, or 31 deferred tokens with one decode token, giving 8, 16, or 32 total token rows. The per-step token budget controls the TPOT increase; smaller captured shapes handle the remaining deferred tokens. Larger decode batches leave less compute capacity for concurrent deferred prefill. Appendix A.3 details graph capture and cache layouts.

If generation ends before deferred prefill finishes, RAINCHECK returns the answer and continues the remaining chunks during available scheduler gaps. Completion resumes from the input checkpoint, adds deferred KV entries, and corrects the live recurrent state without restarting input prefill. The evaluated policy completes all deferred tokens before a later turn reuses the context. An early follow-up waits for completion; Appendix A.3 describes optional bounded recovery.

3 EXPERIMENTS

We test whether deferred prefill recovers answer quality under matched initial token selections, reduces first-turn TTFT with a controlled TPOT increase, and retains quality gains across initial retention ratios and chunk sizes.

3.1 EXPERIMENTAL SETUP

Models and comparisons. We evaluate Qwen3-VL-8B and Qwen3.5-9B, covering full-attention and hybrid GDN architectures (Qwen Team, 2025; 2026). We compare full prefill (`Full`) with hard pruning (`Hard`) and Raincheck (`Ours`) at 30% initial retention. `Hard` denotes existing pruning methods run without token recovery, not a separate algorithm; `Ours` adds deferred prefill to the same initial selection. We adapt MMTok, HiPrune, DivPrune, and VisionZip-Select, marked †, and configure each `Hard`/`Ours` comparison with the same inputs, selection settings, and sampling seeds (Dong et al., 2026; Liu et al., 2025; Alvar et al., 2025; Yang et al., 2025a).

Datasets and scoring. Image and video dialogue benchmarks test whether recovered context supports later questions. Single-turn benchmarks cover visual perception, scene-text understanding, and scientific and mathematical reasoning to assess the effect on the first answer. MultiVerse contains 647 image-grounded dialogues with 2,536 scored turns (Lee et al., 2025); DeepSeek-V4.1-Flash evaluates checklist satisfaction and answer quality using the image and causal dialogue history. ConvBench contains 577 three-turn image dialogues (Liu et al., 2024a); DeepSeek-V4-Flash-FP8 compares each answer with a reference in both presentation orders, conditioned on a fixed caption and the current question. MT-Video-Bench (Pan et al., 2026) uses 902 dialogues from 119 available videos; DeepSeek-V4-Flash-FP8 scores answer-level checklist coverage. MultiVerse aggregates all scored turns, ConvBench averages three turns per dialogue, and video scores average equally across six categories. Table 9 summarizes task coverage; Appendices C.5 and C.3 specify judge instructions and video aggregation. Appendix D.1 reports single-turn results.

Execution and timing. Main quality comparisons use Transformers (Wolf et al., 2020) with 31-token restoration chunks; the main latency comparisons use SGLang on one NVIDIA H20 GPU per worker with one running request and at most 31 deferred tokens per step. Both backends complete the prefix before later-turn reuse. Timing uses MMTok on 48 dialogues per dataset and replays identical output-token traces across policies: 512-token traces on MultiVerse and model-specific `Full` traces on MT-Video-Bench. We measure request-to-first-token TTFT, request-average inter-token TPOT, and end-to-end (E2E) latency through the final output token, excluding post-response restoration. Appendices C.2 and E.1 give generation parameters and timing protocols.

Reporting conventions. Quality scores range from 0 to 100; differences are score points. For metric value m_p under policy p, parentheses report $100(m_p/m_{\text{Full}} - 1)\%$ against the matching model, dataset, turn, and metric under `Full`. Appendix D.3 defines rounding, quality-loss recovery, and paired intervals.

3.2 MULTI-TURN QUALITY RECOVERY

Table 2: Multi-turn quality at 30% initial retention. Parentheses show relative changes from `Full`; bold marks the higher score in each `Hard`/`Ours` pair. Avg and Mean follow the aggregation in Section 3.1.

Model	Pruning method	Policy	MultiVerse				ConvBench			
			T1	T2	T3	Avg	T1	T2	T3	Mean
Qwen3-VL-8B	N/A	`Full`	47.68	53.81	58.11	55.43	50.95	54.94	25.56	43.82
	MMTok†	`Hard`	32.78 (-31.3%)	46.92 (-12.8%)	55.20 (-5.0%)	48.77 (-12.0%)	40.55 (-20.4%)	48.09 (-12.5%)	**25.65** (+0.4%)	38.10 (-13.1%)
		`Ours`	**45.97** (-3.6%)	**53.39** (-0.8%)	**58.96** (+1.5%)	**55.22** (-0.4%)	**47.05** (-7.7%)	**53.64** (-2.4%)	24.78 (-3.1%)	**41.83** (-4.5%)
	HiPrune†	`Hard`	42.37 (-11.1%)	**52.13** (-3.1%)	**58.52** (+0.7%)	53.67 (-3.2%)	45.93 (-9.9%)	51.99 (-5.4%)	24.09 (-5.8%)	40.67 (-7.2%)
		`Ours`	**46.92** (-1.6%)	51.85 (-3.6%)	58.39 (+0.5%)	**55.13** (-0.5%)	**49.65** (-2.6%)	**53.81** (-2.1%)	**24.18** (-5.4%)	**42.55** (-2.9%)
	DivPrune†	`Hard`	36.50 (-23.4%)	48.83 (-9.3%)	56.27 (-3.2%)	50.84 (-8.3%)	44.63 (-12.4%)	50.17 (-8.7%)	**24.70** (-3.4%)	39.83 (-9.1%)
		`Ours`	**45.30** (-5.0%)	**52.46** (-2.5%)	**58.16** (+0.1%)	**54.74** (-1.2%)	**47.23** (-7.3%)	**51.47** (-6.3%)	23.05 (-9.8%)	**40.58** (-7.4%)
	VisionZip-Select†	`Hard`	41.11 (-13.8%)	50.65 (-5.9%)	57.49 (-1.1%)	52.80 (-4.7%)	46.36 (-9.0%)	50.17 (-8.7%)	23.74 (-7.1%)	40.09 (-8.5%)
		`Ours`	**46.70** (-2.1%)	**52.90** (-1.7%)	**58.42** (+0.5%)	**55.07** (-0.6%)	**49.39** (-3.1%)	**52.86** (-3.8%)	**25.13** (-1.7%)	**42.46** (-3.1%)
Qwen3.5-9B	N/A	`Full`	51.35	54.52	41.29	44.90	50.78	55.81	27.12	44.57
	MMTok†	`Hard`	30.48 (-40.6%)	47.49 (-12.9%)	**40.07** (-3.0%)	37.53 (-16.4%)	37.26 (-26.6%)	47.57 (-14.8%)	21.75 (-19.8%)	35.53 (-20.3%)
		`Ours`	**47.41** (-7.7%)	**54.14** (-0.7%)	39.87 (-3.4%)	**43.12** (-4.0%)	**49.91** (-1.7%)	**55.98** (+0.3%)	**25.48** (-6.0%)	**43.79** (-1.8%)
	HiPrune†	`Hard`	41.12 (-19.9%)	51.60 (-5.4%)	40.03 (-3.1%)	41.59 (-7.4%)	47.23 (-7.0%)	50.61 (-9.3%)	23.57 (-13.1%)	40.47 (-9.2%)
		`Ours`	**48.98** (-4.6%)	**53.56** (-1.8%)	**41.27** (0.0%)	**44.05** (-1.9%)	**48.35** (-4.8%)	**55.29** (-0.9%)	**25.56** (-5.8%)	**43.07** (-3.4%)
	DivPrune†	`Hard`	36.49 (-28.9%)	50.07 (-8.2%)	**40.79** (-1.2%)	39.71 (-11.6%)	46.19 (-9.0%)	51.82 (-7.1%)	**25.82** (-4.8%)	41.28 (-7.4%)
		`Ours`	**46.43** (-9.6%)	**52.99** (-2.8%)	40.67 (-1.5%)	**42.90** (-4.5%)	**49.83** (-1.9%)	**54.51** (-2.3%)	24.87 (-8.3%)	**43.07** (-3.4%)
	VisionZip-Select†	`Hard`	40.83 (-20.5%)	51.30 (-5.9%)	**41.39** (+0.2%)	41.28 (-8.1%)	46.19 (-9.0%)	52.51 (-5.9%)	23.92 (-11.8%)	40.87 (-8.3%)
		`Ours`	**47.79** (-6.9%)	**54.11** (-0.8%)	40.72 (-1.4%)	**43.42** (-3.3%)	**50.52** (-0.5%)	**54.59** (-2.2%)	**24.87** (-8.3%)	**43.33** (-2.8%)

Recovering quality at a fixed initial budget. `Ours` improves MultiVerse aggregate quality for all eight model and pruning-method pairs under identical initial selections in Table 2. With MMTok, `Ours` raises the MultiVerse aggregate from 48.77 to 55.22 on Qwen3-VL-8B and from 37.53 to 43.12 on Qwen3.5-9B, approaching `Full` scores of 55.43 and 44.90. Across the eight pairs, `Ours` recovers an average 77.7% of the `Full`-to-`Hard` score loss. Appendix D.3 distinguishes cell-level uncertainty from paired score differences.

Recovery during generation and later reuse. Quality gains begin during first-turn generation and extend to later turns that reuse the completed prefix. For MMTok, the T1 gains are 13.19 and 16.93 points on Qwen3-VL-8B and Qwen3.5-9B, respectively; T2 gains are 6.47 and 6.65 points. Individual turns can regress: Qwen3-VL with HiPrune scores 58.39 under `Ours` at T3, below 58.52 under `Hard`, despite improving the aggregate. Deferred prefill preserves emitted tokens and historical KV entries, so completing the input does not reproduce `Full`'s generation trajectory. Appendix D.6 isolates restoration timing by comparing recovery during generation with recovery only after the first answer.

Complementary evaluation on ConvBench. The same 30%-retention comparison improves the ConvBench mean for all eight pairs by 0.75 to 8.26 points. MMTok scores rise from 38.10 to 41.83 on Qwen3-VL-8B and from 35.53 to 43.79 on Qwen3.5-9B. ConvBench scores measure judge preference against turn-specific reference answers. The `Full` T3 scores are 25.56 and 27.12; cross-turn differences also reflect changes in questions and references. Table 23 reports context-adherence and complete-dialogue comparisons across restoration chunk sizes.

Table 3: MT-Video-Bench quality with MMTok at 30% initial retention. Overall includes all annotated turns; parentheses and bold follow Table 2.

Model	Policy	T1	T2	T3	Overall
Qwen3-VL-8B	`Full`	25.74	59.15	48.78	53.15
	`Hard`	**25.08** (-2.6%)	**57.44** (-2.9%)	46.26 (-5.2%)	50.68 (-4.6%)
	`Ours`	24.92 (-3.2%)	56.60 (-4.3%)	**46.88** (-3.9%)	**51.73** (-2.7%)
Qwen3.5-9B	`Full`	25.66	47.17	39.96	44.28
	`Hard`	26.07 (+1.6%)	45.14 (-4.3%)	41.19 (+3.1%)	42.87 (-3.2%)
	`Ours`	**26.48** (+3.2%)	**47.29** (+0.3%)	**41.44** (+3.7%)	**44.13** (-0.3%)

Multi-turn video quality. `Ours` improves overall video quality on both models in Table 3. Qwen3-VL improves from 50.68 to 51.73, approaching `Full` at 53.15; Qwen3.5 improves from 42.87 to 44.13, close to `Full` at 44.28. Overall includes all annotated turns and weights the six categories equally; the first three columns summarize different category mixtures for each turn. On Qwen3-VL, `Ours` trails `Hard` by 0.16 and 0.84 points on T1 and T2, but gains 0.62 points on T3. Examples from the separate 64-token configuration in Appendix D.4 contrast additional explanations that raise checklist scores with omitted visual details.

3.3 Inference Efficiency

First-turn response latency. `Ours` lowers first-turn TTFT on both datasets, with different decoding tradeoffs in Table 4. On MultiVerse, `Ours` reduces T1 TTFT by 24.1% and 17.9% on Qwen3-VL-8B and Qwen3.5-9B, with mean TPOT increases of 0.13% and 0.47%. The paired `Ours`-minus-`Hard` TTFT intervals include zero in Appendix E.1. On MT-Video-Bench, `Ours` reduces TTFT by 13.06% and 11.91%, and increases average TPOT by 19.93% and 14.92%. Despite the slower decode intervals, mean E2E latency decreases by 4.83% and 6.56% relative to `Full`. `Hard` completes answers sooner than `Ours` but permanently discards the unselected visual tokens. The image and video traces have different lengths, so average TPOT depends on both the latency of decode steps with deferred prefill and the fraction of such steps.

Table 4: Mean first-turn inference latency with matched outputs. Parentheses show relative changes from `Full`.

Policy	TTFT (ms)	TPOT (ms)	E2E (s)
MultiVerse: Qwen3-VL-8B			
`Full`	169.94	6.1919	3.3340
`Hard`	133.17 (-21.64%)	6.1354 (-0.91%)	3.2684 (-1.97%)
`Ours`	128.96 (-24.11%)	6.1996 (+0.13%)	3.2970 (-1.11%)
MultiVerse: Qwen3.5-9B			
`Full`	171.20	6.4934	3.4893
`Hard`	142.31 (-16.87%)	6.4624 (-0.48%)	3.4446 (-1.28%)
`Ours`	140.56 (-17.90%)	6.5238 (+0.47%)	3.4742 (-0.43%)
MT-Video-Bench: Qwen3-VL-8B			
`Full`	3556.18	6.6798	4.8653
`Hard`	3088.11 (-13.16%)	6.3123 (-5.50%)	4.3273 (-11.06%)
`Ours`	3091.61 (-13.06%)	8.0110 (+19.93%)	4.6305 (-4.83%)
MT-Video-Bench: Qwen3.5-9B			
`Full`	3484.92	6.6599	4.4563
`Hard`	3068.06 (-11.96%)	6.4956 (-2.47%)	4.0153 (-9.89%)
`Ours`	3069.86 (-11.91%)	7.6536 (+14.92%)	4.1640 (-6.56%)

Restoration during and after the answer. All MultiVerse requests finish restoration within the measured 512-token traces. The video cohort has about ten times as many deferred tokens and shorter answer traces; 41.67% and 68.75% of video observations require post-response completion on Qwen3-VL and Qwen3.5, respectively. Table 5 reports TPOT during restoration and the time from answer completion to prefix readiness. The video post-response interval averages 74.0 and 162.8 ms over all requests, including those already restored. The remaining prefill can finish before the next request arrives. An earlier follow-up waits for completion before reusing the prefix. Appendix E.1 specifies interval aggregation and later-turn reuse results. Table 7 separates the cache and auxiliary storage required by deferred prefill from the memory used by `Full` and `Hard`.

Table 5: Deferred-prefill completion on 512-token MultiVerse traces and naturally terminated `Full` video traces replayed across policies. Restoration TPOT includes only deferred-prefill intervals; post-response statistics include all first-turn requests, including those already fully restored.

Model	Dataset	TPOT during restoration (ms)	Unfinished at answer completion	Post-response time (ms)	
				Mean	P95
Qwen3-VL-8B	MultiVerse	7.201	0/288 (0.00%)	0.049	0.060
	MT-Video-Bench	8.685	120/288 (41.67%)	74.024	357.703
Qwen3.5-9B	MultiVerse	7.882	0/288 (0.00%)	0.089	0.100
	MT-Video-Bench	8.064	198/288 (68.75%)	162.802	371.782

3.4 Ablation Studies

We test whether quality gains depend on initial retention or deferred-prefill chunk size using matched MMTok sweeps. Appendix D.5 specifies the populations, evaluation, and paired intervals; each sweep uses its own matched baseline.

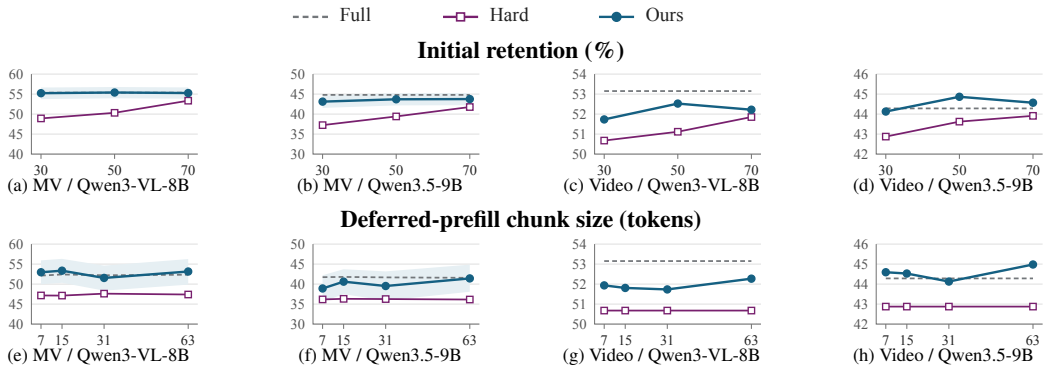


Figure 9: Quality across initial retention ratios and deferred-prefill chunk sizes with MMTok. MV denotes MultiVerse; Video denotes MT-Video-Bench. Panels use separate vertical scales; MultiVerse shading shows 95% marginal confidence intervals.

Initial retention. Restoration improves the aggregate score at each tested retention ratio in the upper row of Figure 9, using 31-token chunks. Lower initial retention generally leaves more score loss to recover. Recovery gains are nonmonotonic in the Qwen3-VL-8B video sweep, where 50% retention yields a larger gain than 30%.

Deferred-prefill chunk size. Quality gains persist across the tested chunk sizes $c \in \{7, 15, 31, 63\}$ at 30% retention in the lower row of Figure 9. On MultiVerse, `Ours` improves over `Hard` in all eight model and chunk comparisons by 2.73 to 6.23 points, and six paired confidence intervals are above zero. On MT-Video-Bench, the `Ours` score spans only 0.54 points on Qwen3-VL-8B and 0.85 points on Qwen3.5-9B across the plotted settings. The 64-token video setting scores 1.32 points above the 31-token setting on Qwen3.5-9B, with a paired interval excluding zero in Appendix D.5. Chunk size therefore affects both quality recovery and TPOT during deferred prefill, as measured in Figure 5.

4 Related Work

Multimodal token pruning reduces the tokens processed by the model through different pruning locations and selection criteria. To preserve information from pruned tokens, some methods aggregate token features or learn compressed representations (Zhang et al., 2025b; Yang et al., 2025a; Wang et al., 2026). Other methods retain candidate tokens or intermediate states for later retrieval, reselection, or on-demand computation (Khaki et al., 2025; Kim et al., 2026; Fu et al., 2024). RainCheck separates initial selection from recovery by adding deferred input states during generation and after answer return for later reuse. Attention visibility control and recurrent state mapping support insertion before an already processed suffix. See Appendix H for details.

5 Conclusion

RainCheck addresses a central tension in latency-sensitive multimodal interaction: answering quickly without permanently losing the input skipped to make that possible. It processes selected tokens up front and recovers deferred tokens during generation and after answer return. Attention visibility control and affine state mapping make the recovered context usable for subsequent decoding, while fused execution shares weight access between prefill and decode. Across image and video dialogues on both evaluated models, RainCheck improves aggregate quality over matched hard pruning. On MultiVerse, it reduces mean first-turn TTFT by up to 24.1% relative to full prefill, with less than a 0.5% increase in mean TPOT. Our single-request evaluation establishes this opportunity for latency-sensitive inference; concurrency and state-mapping assumptions remain open questions (Appendix G.1). We hope this work inspires consumer-facing multimodal systems that defer computation to respond quickly, without making lost context the price of responsiveness.

AI USE STATEMENT

Generative AI tools were used for language editing and code assistance. The authors verified all statements, derivations, citations, and experimental artifacts and take full responsibility.

REFERENCES

Amey Agrawal, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, and Ramachandran Ramjee. SARATHI: Efficient LLM inference by piggybacking decodes with chunked prefills. arXiv preprint arXiv:2308.16369, 2023. URL https://arxiv.org/abs/2308.16369.

Amey Agrawal, Nitin Kedia, Ashish Panwar, Jayashree Mohan, Nipun Kwatra, Bhargav S. Gulavani, Alexey Tumanov, and Ramachandran Ramjee. Taming throughput-latency tradeoff in LLM inference with Sarathi-Serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024. URL https://arxiv.org/abs/2403.02310.

Saeed Ranjbar Alvar, Gursimran Singh, Mohammad Akbari, and Yong Zhang. DivPrune: Diversity-based visual token pruning for large multimodal models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9392–9401, 2025. URL https://arxiv.org/abs/2503.02175.

Liang Chen, Haozhe Zhao, Tianyu Liu, Shuai Bai, Junyang Lin, Chang Zhou, and Baobao Chang. An image is worth 1/2 tokens after layer 2: Plug-and-play inference acceleration for large vision-language models. In *European Conference on Computer Vision*, 2024. URL https://arxiv.org/abs/2403.06764.

DeepSeek-AI. DeepSeek-V4.1-Flash: Pushing the limits of KV cache compression. Hugging Face model card, 2026a. URL https://huggingface.co/deepseek-ai/DeepSeek-V4.1-Flash.

DeepSeek-AI. DeepSeek-V4-Flash. Hugging Face model card, 2026b. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash.

Sixun Dong, Juhua Hu, Mian Zhang, Ming Yin, Yanjie Fu, and Qi Qian. MMTok: Multimodal coverage maximization for efficient inference of VLMs. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=GvPdSWZT31.

Qichen Fu, Minsik Cho, Thomas Merth, Sachin Mehta, Mohammad Rastegari, and Mahyar Najibi. LazyLLM: Dynamic token pruning for efficient long context LLM inference. arXiv preprint arXiv:2407.14057v1, 2024. URL https://arxiv.org/abs/2407.14057v1.

Tianxiao Gao, Shanwei Zhao, Shuo Fang, Shiai Zhu, and Chenguang Ma. QuietPrune: Query-guided early token pruning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3553–3562, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Gao_QuietPrune_Query-Guided_Early_Token_Pruning_for_Vision-Language_Models_CVPR_2026_paper.html.

Haoyang Huang, Wenjie Huang, Tianqi Xu, Hongyaoxing Gu, Kang Tan, Yikai Fu, Yuhao Shen, Tianyu Liu, Baolin Zhang, Jun Zhang, Xinyi Hu, Jun Dai, Shuang Ge, Lei Chen, Yue Li, Mingchen Wang, and Meng Zhang. OmniDelta: Skill-driven budget allocation for token compression in OmniLLMs. arXiv preprint arXiv:2607.25669, 2026. URL https://arxiv.org/abs/2607.25669.

Drew A. Hudson and Christopher D. Manning. GQA: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6700–6709, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Hudson_GQA_A_New_Dataset_for_Real-World_Visual_Reasoning_and_Compositional_CVPR_2019_paper.html.

Yicheng Ji, Jun Zhang, Heming Xia, Jinpeng Chen, Lidan Shou, Gang Chen, and Huan Li. SpecVLM: Enhancing speculative decoding of video LLMs via verifier-guided token pruning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 7205–7219, 2025. doi: 10.18653/v1/2025.emnlp-main.366. URL https://aclanthology.org/2025.emnlp-main.366/.

Yicheng Ji, Jun Zhang, Jinpeng Chen, Cong Wang, Lidan Shou, Gang Chen, and Huan Li. See the forest for the trees: Loosely speculative decoding via visual-semantic guidance for efficient inference of video LLMs. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 23707–23726, 2026. doi: 10.18653/v1/2026.acl-long.1087. URL https://aclanthology.org/2026.acl-long.1087/.

Aditya K. Kamath, Ramya Prabhu, Jayashree Mohan, Simon Peter, Ramachandran Ramjee, and Ashish Panwar. POD-Attention: Unlocking full prefill-decode overlap for faster LLM inference. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 2025. doi: 10.1145/3676641.3715996.

Samir Khaki, Junxian Guo, Jiaming Tang, Shang Yang, Yukang Chen, Konstantinos N. Plataniotis, Yao Lu, Song Han, and Zhijian Liu. SparseVILA: Decoupling visual sparsity for efficient VLM inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 23784–23794, 2025. URL https://arxiv.org/abs/2510.17777.

Jiwan Kim, Kibum Kim, Wonjoong Kim, Byung-Kwan Lee, and Chanyoung Park. Why and when visual token pruning fails? a study on relevant visual information shift in MLLMs decoding. In *European Conference on Computer Vision*, 2026. URL https://arxiv.org/abs/2604.12358.

Kimi Team. Kimi Linear: An expressive, efficient attention architecture. arXiv preprint arXiv:2510.26692, 2025. URL https://arxiv.org/abs/2510.26692.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. URL https://arxiv.org/abs/2309.06180.

Young-Jun Lee, Byung-Kwan Lee, Jianshu Zhang, Yechan Hwang, Byungsoo Ko, Han-Gyu Kim, Dongyu Yao, Xuankun Rong, Eojin Joo, Seung-Ho Han, et al. MultiVerse: A multi-turn conversation benchmark for evaluating large vision and language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025. URL https://arxiv.org/abs/2510.16641.

Ao Li, Yuxiang Duan, Jinghui Zhang, Congbo Ma, Yutong Xie, Gustavo Carneiro, Mohammad Yaqub, and Hu Wang. TransPrune: Token transition pruning for efficient large vision-language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 39529–39538, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Li_TransPrune_Token_Transition_Pruning_for_Efficient_Large_Vision-Language_Model_CVPR_2026_paper.html.

Jizhihui Liu, Feiyi Du, Guangdao Zhu, Niu Lian, Jun Li, and Bin Chen. HiPrune: Training-free visual token pruning via hierarchical attention in vision-language models. arXiv preprint arXiv:2508.00553v1, 2025. URL https://arxiv.org/abs/2508.00553v1.

Shuo Liu, Kaining Ying, Hao Zhang, Yue Yang, Yuqi Lin, Tianle Zhang, Chuanhao Li, Yu Qiao, Ping Luo, Wenqi Shao, and Kaipeng Zhang. ConvBench: A multi-turn conversation evaluation benchmark with hierarchical ablation capability for large vision-language models. In *Advances in Neural Information Processing Systems*, 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/b69396afc07a9ca3428d194f4db84c02-Abstract-Datasets_and_Benchmarks_Track.html.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. MMBench: Is your multi-modal model an all-around player? In *European Conference on Computer Vision*, 2024b. URL https://eccv.ecva.net/virtual/2024/poster/2003.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/11332b6b6cf4485b84afadb1352d3a9a-Abstract-Conference.html.

Yaning Pan, Qianqian Xie, Guohui Zhang, Zekun Moore Wang, Yongqian Wen, Yuanxing Zhang, Haoxuan Hu, Zhiyu Pan, Yibing Huang, Zhidong Gan, Yonghong Lin, An Ping, Shihao Li, Yanghai Wang, Tianhao Peng, and Jiaheng Liu. MT-Video-Bench: A holistic video understanding benchmark for evaluating multimodal LLMs in multi-turn dialogues. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8105–8126, 2026. doi: 10.18653/v1/2026.findings-acl.397. URL https://aclanthology.org/2026.findings-acl.397/.

Qwen Team. Qwen3-VL technical report. *arXiv preprint arXiv:2511.21631*, 2025. URL https://arxiv.org/abs/2511.21631.

Qwen Team. Qwen3.5. Hugging Face model card, 2026. URL https://huggingface.co/Qwen/Qwen3.5-9B.

Chaoyi Ruan, Yinhe Chen, Dongqi Tian, Yandong Shi, Yongji Wu, Jialin Li, and Cheng Li. Libra: Flexible request partitioning and scheduling for serving unbalanced and dynamic LLM workloads. In *23rd USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2026. URL https://www.usenix.org/conference/nsdi26/presentation/ruan-libra.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/7ede97c3e082c6df10a8d6103a2eebd2-Abstract-Conference.html.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8317–8326, 2019. URL https://openaccess.thecvf.com/content_CVPR_2019/html/Singh_Towards_VQA_Models_That_Can_Read_CVPR_2019_paper.html.

SyFI TraceLab. SyFI TraceLab: Tool latency distribution. Public coding-agent timing analysis, 2026. URL https://tracelab.cs.washington.edu/exp/tool_calls/tool_latency_distribution/. Accessed September 14, 2026.

Keda Tao, Can Qin, Haoxuan You, Yang Sui, and Huan Wang. DyCoke: Dynamic compression of tokens for fast video large language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://arxiv.org/abs/2411.15024.

Yi Wang, Haofei Zhang, Qihan Huang, Anda Cao, Gongfan Fang, Wei Wang, Xuan Jin, Jie Song, Mingli Song, and Xinchao Wang. Rethinking token reduction for large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24727–24737, 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Wang_Rethinking_Token_Reduction_for_Large_Vision-Language_Models_CVPR_2026_paper.html.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural

language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Long Xing, Qidong Huang, Xiaoyi Dong, Jiajie Lu, Pan Zhang, Yuhang Zang, Yuhang Cao, Conghui He, Jiaqi Wang, Feng Wu, and Dahua Lin. Conical visual concentration for efficient large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/papers/Xing_Conical_Visual_Concentration_for_Efficient_Large_Vision-Language_Models_CVPR_2025_paper.pdf.

Yueru Yan, Tuc Nguyen, Bo Su, Melissa Lieffers, and Thai Le. ShareChat: A dataset of chatbot conversations in the wild. arXiv preprint arXiv:2512.17843v4, 2026. URL https://arxiv.org/abs/2512.17843v4.

Senqiao Yang, Yukang Chen, Zhuotao Tian, Chengyao Wang, Jingyao Li, Bei Yu, and Jiaya Jia. VisionZip: Longer is better but not necessary in vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19792–19802, 2025a. URL https://arxiv.org/abs/2412.04467.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=r8H7xhYPwz.

Xubing Ye, Yukang Gan, Yixiao Ge, Xiao-Ping Zhang, and Yansong Tang. ATP-LLaVA: Adaptive token pruning for large vision language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 24972–24982, 2025. URL https://openaccess.thecvf.com/content/CVPR2025/html/Ye_ATP-LLaVA_Adaptive_Token_Pruning_for_Large_Vision_Language_Models_CVPR_2025_paper.html.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, Qin Xie, Guiming Xie, and Xuejian Gong. HMI: hierarchical knowledge management for efficient multi-tenant inference in pretrained language models. *The VLDB Journal*, 34(4):43, 2025a. doi: 10.1007/s00778-025-00919-7.

Qizhe Zhang, Aosong Cheng, Ming Lu, Zhiyong Zhuo, Minqi Wang, Jiajun Cao, Shaobo Guo, Qi She, and Shanghang Zhang. [CLS] attention is all you need for training-free visual token pruning: Make VLM inference faster. arXiv preprint arXiv:2412.01818v1, 2024a. URL https://arxiv.org/abs/2412.01818v1.

Renrui Zhang, Dongzhi Jiang, Yichi Zhang, Haokun Lin, Ziyu Guo, Pengshuo Qiu, Aojun Zhou, Pan Lu, Kai-Wei Chang, Peng Gao, and Hongsheng Li. MathVerse: Does your multi-modal LLM truly see the diagrams in visual math problems? In *European Conference on Computer Vision*, 2024b. URL https://eccv.ecva.net/virtual/2024/poster/1852.

Yuan Zhang, Chun-Kai Fan, Junpeng Ma, Wenzhao Zheng, Tao Huang, Kuan Cheng, Denis Gudovskiy, Tomoyuki Okuno, Yohei Nakata, Kurt Keutzer, and Shanghang Zhang. SparseVLM: Visual token sparsification for efficient vision-language model inference. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 74840–74857, 2025b. URL https://proceedings.mlr.press/v267/zhang25s.html.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, 2024. URL https://arxiv.org/abs/2312.07104.

Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, and Hao Zhang. DistServe: Disaggregating prefill and decoding for goodput-optimized large language model serving. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024. URL https://arxiv.org/abs/2401.09670.

A Notation and Implementation Details

A.1 Notation and Figure Conventions

Table 6 expands Table 1. Token symbols denote identities or ordered sequences, not embedding vectors. Decode index t is distinct from dialogue turns T1, T2, and T3. Subscripts $< t$ and $\leq t$ mean before and through step t. A prime marks deferred insertion at a fixed generated-sequence endpoint. GDN shapes refer to one layer and one head; both indices are omitted. Vectors are columns unless transposed.

Table 6: Complete notation with object types and tensor shapes.

Symbol	Type / shape	Definition
Input and execution		
X, K, D	Token sequences	Targeted multimodal tokens; important and deferred partitions in source order.
$\mathcal{T}_{\text{tail}}$	Token sequence	Processed input between the insertion boundary and generated tokens, including text, control, or special tokens.
$\mathcal{T}, I_0$	Token sequences	All fixed text/control/special tokens; initial input combining $\mathcal{T}$ and K in source order.
N_m, b_m, ρ	Counts; scalar ratio	Span size, initial retained count, deferred fraction; $b_m = N_m - \lfloor \rho N_m \rfloor$.
I_t	Token sequence	Materialized input before step t: $I_t = I_0 \Vert D_{<t}$.
$G_t, g_{t,j}, c$	Sequence; token; count	Deferred group, its jth token, maximum deferred tokens per step.
$D_{<t}, D_{\leq t}$	Token sequences	Deferred prefix before/through t; $D_{\leq t} = D_{<t} \Vert G_t$.
$y_t, Y_{<t}, Y_{\leq t}, Y$	Token; sequences	Decode input token and processed generated history. Emitted tokens enter Y only after their forward executes.
$\Vert, \mathcal{V}(x)$	Operator; set	Sequence concatenation; token identities visible to query token x.
d_a	Positive integer	Full-attention head dimension.
KV_U	Pair in $\mathbb{R}^{\lvert U \rvert \times d_a}$	Per-layer/head key and value matrices for sequence U.
GDN state and affine transition		
d_k, d_v	Positive integers	Key and value dimensions of one GDN head.
$S, S_{\text{prefix}}, S_{\text{live}}$	$\mathbb{R}^{d_k \times d_v}$	S is a recurrent state. S_{prefix} ends at the recovery boundary; S_{live} follows the processed suffix $\mathcal{T}_{\text{tail}} \Vert Y$.
$\bar{S} = S^\top$	$\mathbb{R}^{d_v \times d_k}$	Value-by-key state orientation used by the implementation.
$\mathbf{k}_x, \mathbf{v}_x$	$\mathbb{R}^{d_k}$; $\mathbb{R}^{d_v}$	Key and value column vectors after short convolution and activation; the key is normalized.
α_x, β_x	Scalars	Decay and delta-rule update gates.
$\mathbf{I}_{d_k}$	$\mathbb{R}^{d_k \times d_k}$	Identity matrix.
A_x, B_x	$\mathbb{R}^{d_k \times d_k}$; $\mathbb{R}^{d_k \times d_v}$	Linear and additive coefficients of $S_{\text{next}} = A_x S + B_x$.
U, Y	Token sequences	Generic recorded suffix; already processed generated tokens. The main-text suffix is $U = \mathcal{T}_{\text{tail}} \Vert Y$.
F_U, F_Y	Affine functions	Maps from $\mathbb{R}^{d_k \times d_v}$ to $\mathbb{R}^{d_k \times d_v}$; $F_U(S) = P_U S + B_U$.
$P_U, P_{\text{tail}}, P_Y$	$\mathbb{R}^{d_k \times d_k}$	Linear coefficients for the generic suffix, the complete tail, and generated tokens. $P_{\text{tail}} = P_Y P_{\mathcal{T}_{\text{tail}}}$.
$B_U, B_{\text{tail}}, B_Y$	$\mathbb{R}^{d_k \times d_v}$	Accumulated additive coefficients. The incremental join does not store the coefficients separately.
$U_{\text{boundary}}, U_{\text{rest}}, w$	Sequences; count	Bounded leading suffix segment containing convolution-sensitive tokens, remaining suffix, and convolution width.
$S_{\text{anchor}}, S'_{\text{anchor}}$	$\mathbb{R}^{d_k \times d_v}$	States after U_{boundary}, before and after boundary repair.

Table 6: continued.

Symbol	Type / shape	Definition
Block-matrix representation		
$Z(S)$	$\mathbb{R}^{(d_k+d_v)\times d_v}$	State S stacked above $\mathbf{I}_{d_v}$.
M_x, M_U, M_V	$\mathbb{R}^{(d_k+d_v)\times(d_k+d_v)}$	Augmented transitions for one token or a token block; upper blocks contain the linear and additive coefficients.
V, P_V, B_V	Sequence; matrices	Block following U, with $P_V \in \mathbb{R}^{d_k\times d_k}$ and $B_V \in \mathbb{R}^{d_k\times d_v}$.
Generalizations and figure labels		
$P_{\mathcal{T}_{\text{tail}}}, P_Q$	$\mathbb{R}^{d_k\times d_k}$	Transitions through intervening input or an added prompt block Q.
γ, Q	Scalar ratio; sequence	Optional recovery fraction and newly processed prompt tokens.
$S^{\text{compact}}, S^{\text{complete}}, S_{\text{reuse}}$	$\mathbb{R}^{d_k\times d_v}$	Compact, completed, and reusable recurrent endpoints in the PD design.
KV_{reuse}	Pair of matrices	Keys and values after the PD state join; each matrix has $\lvert I_0\rvert + \lvert D\rvert + \lvert Y\rvert$ rows and d_a columns per layer and KV head.
Q_i, A_i	Dialogue messages	Question and answer in the overview and motivation figures.
$t_{Q_1}, t_{Q_2}, t_{A_1}, t_{E_1}, t_{R_1}$	Time points	Question arrivals, first output, answer completion, and prefix completion in the overview.

Reading the method figures. Blue, purple, black, and green denote important multimodal, deferred multimodal, intervening input, and generated tokens. Black tokens include text and control or special tokens; the label does not restrict the intervening sequence to natural-language text. Earlier context is included in the prefix checkpoint without a separate symbol. A recurrent state marks the endpoint of a processed sequence, not the representation of one isolated token. The examples use d_1 as a one-token deferred group and correct the state after y_1 before processing y_2. A scheduled group can contain multiple deferred tokens. The illustrated $P_{\text{tail}} = P_Y P_{\mathcal{T}_{\text{tail}}}$ covers black input and green generated tokens. The evaluated layout has $\mathcal{T}_{\text{tail}} = \varnothing$, so $P_{\text{tail}} = P_Y$; Appendix A.2 specifies the cache layout.

A.2 Sequence and Checkpoint Conventions

The evaluated implementation inserts deferred tokens after all fixed input tokens and before the processed generated tokens. Let X contain the targeted multimodal tokens and $\mathcal{T}$ contain all non-targeted text, control, and special tokens. A pruning method partitions X into selected tokens K and deferred tokens D. For span m with N_m tokens and deferred fraction ρ, the initial budget is $b_m = N_m - \lfloor \rho N_m \rfloor$. The retained multimodal tokens and $\mathcal{T}$ form I_0 in source order. Deferred tokens also preserve source order across spans and chunks. Before decode step t, the materialized input is

$$I_t = I_0 \| D_{<t}, \tag{3}$$

where $D_{<t}$ contains already processed deferred tokens and $\|$ denotes concatenation. The execution sequence is $I_t \| Y_{<t}$, and the input checkpoint follows I_t. The evaluated boundary has $\mathcal{T}_{\text{tail}} = \varnothing$, so $P_{\text{tail}} = P_Y$ in Equation 1.

For group $G_t = (g_{t,1}, \ldots, g_{t,|G_t|})$, the two attention views are

$$\begin{aligned} \mathcal{V}(g_{t,j}) &= I_t \cup \{g_{t,1}, \ldots, g_{t,j}\}, \\ \mathcal{V}(y_t) &= I_t \cup G_t \cup Y_{<t} \cup \{y_t\}. \end{aligned} \tag{4}$$

Here $\mathcal{V}(x)$ denotes the visible token identities; positions distinguish repeated occurrences of the same token. After the step, $D_{\le t} = D_{<t} \| G_t$ and $Y_{\le t} = Y_{<t} \| y_t$. The valid physical cache regions are

$$\text{KV}_t = [\text{KV}_{I_0} \mid \text{KV}_{D_{\le t}} \mid \text{KV}_{Y_{\le t}}]. \tag{5}$$

For a sequence U, KV_U denotes a pair of matrices in $\mathbb{R}^{|U|\times d_a}$ per layer and KV head, where d_a is the full-attention head dimension. Pending deferred capacity between the materialized input and Y contains no readable state. Original positions supply position encodings; physical regions determine storage; the schedule determines availability. The cache never uses original positions as sparse write addresses.

Insertion-boundary convention. The method figures show a nonempty intervening input sequence $\mathcal{T}_{\mathrm{tail}}$ between the deferred region and Y. The black input tokens can include text, control markers, or other token types outside the deferred partition. Earlier context is part of the prefix checkpoint and requires no separate symbol. Moving the insertion boundary changes both deferred attention visibility and the suffix covered by P_{tail}. For an empty intervening sequence, $P_{\mathcal{T}_{\mathrm{tail}}}$ is the identity matrix, recovering the layout in Equation 5.

A.3 Execution Details

Pruning-method adapters. The Transformers backend adapts public pruning methods to produce selected original-token indices. Each adapter extracts Qwen features and applies the corresponding selection objective and budget. MMTok and HiPrune use model-specific Qwen adaptations, and DivPrune uses an approximate feature adaptation. VisionZip-Select retains original-token representatives and omits embedding merging. The execution interface consumes the resulting partition and does not require a shared scoring function.

Fused execution and cache storage. Both backends process $[G_t \mid y_t]$ in one pass through the decoder. Normalization, dense projections, output projection, and the feed-forward network operate on the combined sequence. Full-attention layers invoke a causal chunk-attention operation for G_t and a single-token operation for y_t. GDN layers first process the deferred group, then repair the convolution boundary and join the recurrent states using Equation 14, and finally process the decode token. The SGLang recurrent kernel also updates the generated-sequence transition matrix. A serial materialize-then-decode path serves as an implementation control.

The Transformers backend inserts deferred KV rows before the generated suffix in a contiguous buffer. The SGLang backend reserves physical pages for D, so generated KV addresses remain fixed as deferred entries become valid. Both layouts implement Equation 5. Page-table metadata exposes only the processed input and generated regions.

CUDA graph shapes. SGLang captures fused forwards with a fixed number of deferred rows and one decode row. Configurations with 7, 15, and 31 deferred rows produce 8, 16, and 32 total rows for shared dense projections. Fixed shapes permit graph replay with persistent input buffers and reduce repeated launch preparation. Regular matrix dimensions can improve dense-kernel tile utilization.

Additional captured shapes handle smaller budgets and final remainders. At each step, the scheduler selects the largest captured count within both the configured budget and the remaining deferred count. The scheduler processes the remaining deferred tokens in exact-size steps without recurrent updates for padding tokens. A 16-token decode guard preserves the initial pruned context for the first 16 output tokens before deferred prefill begins. The formal systems configuration uses a maximum of 31 deferred tokens per step. The quality experiments use 31-token chunks in the Transformers backend, as specified in Appendix C.2. Chunk size and graph shape are backend settings, not changes to the source-order or state-update rules.

Completion and continuation. Cooperative completion returns the finalized answer before prefilling the remaining deferred tokens. The Transformers backend exposes explicit completion calls; SGLang maintains the deferred state through its scheduler and session objects. Blocking completion finishes deferred prefill before returning the answer and serves as a control. The evaluated full-recovery policy admits a continuation after prefix completion. If a later turn adds media, only the new media is encoded and partitioned. Each new deferred stream receives an input checkpoint and a map over the processed suffix. The backends support one incomplete deferred stream per sequence.

The runtime records answer-ready, response-ready, prefix-ready, and engine-return boundaries. Blocking response-ready latency uses the engine-return boundary after completion; cooperative

response-ready latency uses the earlier answer-return boundary. Appendix E.1 defines the measurement populations and protocols.

Optional bounded recovery. The reported experiments complete recovery before continuation; an optional policy admits a text-only follow-up with unfinished deferred tokens. The scheduler finishes any in-flight state update, admits the new prompt, and resumes pending deferred chunks during later decode steps or scheduling gaps. For an already processed suffix U, a new prompt block Q, including its control tokens, extends the transition matrix as $P_{U\|Q} = P_Q P_U$. Each transition matrix has shape $d_k \times d_k$.

An optional recovery fraction $\gamma \in [0, 1]$ limits materialization to the first $\lfloor \gamma |D| \rfloor$ source-ordered deferred tokens. Pausing retains pending tokens for later processing. Reaching the target or explicitly truncating recovery after a completed state update discards the unmaterialized remainder.

A.4 Memory Accounting

Deferred prefill changes when input states are computed, not the final number of cached tokens under full recovery. Table 7 compares storage for the same input and generated-token count, excluding shared model weights. Let $\mathcal{L}_{\text{attn}}$ contain the full-attention layers. The KV storage per token is $b_{\text{KV}} = 2 s_{\text{KV}} \sum_{\ell \in \mathcal{L}_{\text{attn}}} h_{\text{KV},\ell} d_{\text{head},\ell}$ bytes. Here s_{KV} is bytes per element, and $h_{\text{KV},\ell}$ and $d_{\text{head},\ell}$ are the KV-head count and head dimension of layer ℓ; the factor of two accounts for keys and values. GDN layers store recurrent states rather than per-token KV entries.

Table 7: Component-level allocation requirements for Raincheck relative to the baselines at matched input and output lengths.

Component	Compared with `Full`	Compared with `Hard`
KV after full recovery	Same token count	$\|D\| b_{\text{KV}}$ additional bytes
Deferred input features	Longer residency of cached features; no duplicate allocation required	Same residency extension if features would otherwise be evicted
GDN state mapping	Additional prefix checkpoints, transition matrices, join buffers, and convolution history	Same additional state structures
Scheduling metadata	Deferred indices, progress counters, and cache mappings	Same additional structures
Execution buffers	Fused-step and completion buffers; peak depends on buffer reuse and graph capture	Additional prefill buffers beyond decode-only execution

The valid KV entries grow as deferred tokens are processed, but the SGLang implementation reserves deferred pages in advance to keep generated-token addresses fixed. Reservation therefore does not provide the persistent KV-capacity savings of `Hard`. When the VLM feature cache already holds the encoded input, deferred prefill can reuse the same allocation and delay eviction until the pending tokens are processed. Extending residency changes the release time, not the size of the feature allocation, and need not increase peak GPU memory. Peak memory depends on whether the retained features overlap other allocations; a separate copy would require additional storage.

For each GDN layer and head, the transition matrix has d_k^2 entries and each recurrent checkpoint has $d_k d_v$ entries, independent of decoded-suffix length. The evaluated Qwen3.5-shaped allocation contains 48 MiB of transition matrices within 192 MiB of auxiliary recurrent-state storage across 24 layers, as detailed in Appendix B.6. Convolution buffers, metadata, and CUDA-graph buffers are separate allocations; Table 7 distinguishes cached-feature residency from additional allocations. Full-attention-only models do not require the GDN structures. Component counts alone do not determine peak GPU memory. Peak memory also depends on temporary activations during full prefill and on cache pools and graph buffers that serving engines may preallocate.

B Online Dense State Mapping for GDN

B.1 Single-Token State Update

All matrices and vectors below refer to one layer and one head. The key and value dimensions are d_k and d_v. We use $S \in \mathbb{R}^{d_k \times d_v}$, the transpose of the value-by-key state stored by the implementation.

For a text, multimodal, or generated token x, hold the projected key, value, and gates fixed. The recurrent update is then affine in the incoming state:

$$S_{\text{next}} = A_x S + B_x, \qquad A_x = \alpha_x(\mathbf{I}_{d_k} - \beta_x \mathbf{k}_x \mathbf{k}_x^\top), \qquad B_x = \beta_x \mathbf{k}_x \mathbf{v}_x^\top. \tag{6}$$

The column vectors $\mathbf{k}_x \in \mathbb{R}^{d_k}$ and $\mathbf{v}_x \in \mathbb{R}^{d_v}$ are the key and value after short convolution and activation. The key is normalized before the recurrent update. The scalars α_x and β_x control decay and the delta-rule update (Yang et al., 2025b). Thus $\mathbf{k}_x\mathbf{k}_x^\top$, $\mathbf{I}_{d_k}$, and A_x have shape $d_k \times d_k$; $\mathbf{k}_x\mathbf{v}_x^\top$ and B_x have shape $d_k \times d_v$. The state update scales and modifies the incoming memory before adding the new key-value contribution. In the implementation orientation, $\bar{S} = S^\top \in \mathbb{R}^{d_v \times d_k}$ and $\bar{S}_{\text{next}} = \bar{S}A_x^\top + B_x^\top$.

An augmented matrix expresses the affine update as matrix multiplication. Define the stacked state $Z(S) \in \mathbb{R}^{(d_k+d_v)\times d_v}$ and the transition matrix $M_x \in \mathbb{R}^{(d_k+d_v)\times(d_k+d_v)}$ by

$$Z(S) = \begin{bmatrix} S \\ \mathbf{I}_{d_v} \end{bmatrix}, \qquad M_x = \begin{bmatrix} A_x & B_x \\ 0_{d_v \times d_k} & \mathbf{I}_{d_v} \end{bmatrix}, \qquad Z(S_{\text{next}}) = M_x Z(S). \tag{7}$$

The upper block gives $A_x S + B_x$, and the lower block preserves $\mathbf{I}_{d_v}$. The augmented matrices provide a derivation of affine composition; the implementation maintains the recurrent state and transition coefficients directly.

B.2 Suffix Mapping and State Join

Let $U = (x_1, \ldots, x_r)$ be an ordered sequence of tokens whose transition factors have been recorded. Let F_U map an incoming state to the state after U under the recorded factors. Multiplying the recorded transitions in processing order gives

$$M_U = M_{x_r} \cdots M_{x_1} = \begin{bmatrix} P_U & B_U \\ 0_{d_v \times d_k} & \mathbf{I}_{d_v} \end{bmatrix}, \qquad Z(F_U(S)) = M_U Z(S). \tag{8}$$

The matrix M_U has shape $(d_k + d_v) \times (d_k + d_v)$. The upper block defines the affine function F_U on state matrices:

$$\begin{aligned} F_U &: \mathbb{R}^{d_k \times d_v} \to \mathbb{R}^{d_k \times d_v}, \\ F_U(S) &= P_U S + B_U, \\ P_U &= A_{x_r} \cdots A_{x_1}, \\ B_U &= \sum_{i=1}^{r} (A_{x_r} \cdots A_{x_{i+1}}) B_{x_i}. \end{aligned} \tag{9}$$

$P_U \in \mathbb{R}^{d_k \times d_k}$ and $B_U \in \mathbb{R}^{d_k \times d_v}$ are the linear and additive matrix coefficients of the affine function F_U. The rightmost transition acts first. For example, $F_{(x_1,x_2)}(S) = A_{x_2}A_{x_1}S + A_{x_2}B_{x_1} + B_{x_2}$. The empty sequence has $P_\varnothing = \mathbf{I}_{d_k}$ and $B_\varnothing = 0_{d_k \times d_v}$. The matrix P_U propagates the incoming memory; B_U accumulates token-dependent additions.

For ordered token blocks U followed by V, block multiplication gives

$$\begin{aligned} M_{U\|V} = M_V M_U &= \begin{bmatrix} P_V & B_V \\ 0 & \mathbf{I}_{d_v} \end{bmatrix} \begin{bmatrix} P_U & B_U \\ 0 & \mathbf{I}_{d_v} \end{bmatrix} \\ &= \begin{bmatrix} P_V P_U & P_V B_U + B_V \\ 0 & \mathbf{I}_{d_v} \end{bmatrix}. \end{aligned} \tag{10}$$

Each lower-left zero block has shape $d_v \times d_k$. The later block propagates both the incoming-state coefficient and the additions accumulated by the earlier block. Equation 10 describes composition across token blocks; the online implementation uses the rank-one update in Equation 13.

For two incoming states and the same recorded factors,

$$\begin{aligned} F_U(S') - F_U(S) &= (P_U S' + B_U) - (P_U S + B_U) \\ &= P_U(S' - S). \end{aligned} \tag{11}$$

Every term has shape $d_k \times d_v$. Setting $U = \mathcal{T}_{\text{tail}}\|Y$, $S = S_{\text{prefix}}$, and $S' = S'_{\text{prefix}}$ gives Equation 2, with $P_U = P_{\text{tail}}$. The live state already contains the additive contribution, so incremental correction

stores P_{tail} without separately storing B_{tail}. After correction, the new prefix state becomes the reference for the next deferred update.

Applying Equation 10 to the intervening input followed by generated tokens gives

$$P_U = P_Y P_{\mathcal{T}_{\text{tail}}}, \qquad B_U = P_Y B_{\mathcal{T}_{\text{tail}}} + B_Y. \tag{12}$$

Each transition matrix has shape $d_k \times d_k$, and all additive matrices have shape $d_k \times d_v$. The coefficients in Equation 1 are $P_{\text{tail}} = P_U$ and $B_{\text{tail}} = B_U$. The evaluated input-end boundary has an empty intervening sequence, so $U = Y$ and the online transition starts with the identity matrix.

B.3 Online Update and Storage

Processing generated token y extends the affine map from F_Y to $F_{Y\|y}$: $F_{Y\|y}(S) = A_y F_Y(S) + B_y$. The new linear coefficient is $P_{Y\|y} = A_y P_Y$. Expanding A_y gives

$$P_{Y\|y} = \alpha_y \big(P_Y - \beta_y \mathbf{k}_y \big(\mathbf{k}_y^\top P_Y\big)\big)\,. \tag{13}$$

The row vector $\mathbf{k}_y^\top P_Y$ has shape $1 \times d_k$, so the outer product $\mathbf{k}_y(\mathbf{k}_y^\top P_Y)$ has shape $d_k \times d_k$. The recurrent decode kernel reuses the key and gates to update the transition matrix alongside the live state. Each head stores d_k^2 transition entries regardless of $|Y|$. The recurrent checkpoints each contain $d_k d_v$ entries. The short-convolution boundary buffers also have fixed size.

B.4 Short-Convolution Boundary Handling

For the evaluated input-end insertion boundary, deferred insertion changes the convolution history at the beginning of Y. For convolution width w, at most the first $w-1$ generated-token transitions depend on the changed input history when the pre-convolution projections are held fixed. The executor stores a bounded leading segment containing the convolution-sensitive tokens. Boundary repair reuses the recorded pre-convolution projections and gates, but recomputes the keys, values, and recurrent transitions in the segment from the updated input convolution state.

Write $Y = U_{\text{boundary}} \| U_{\text{rest}}$. The anchors $S_{\text{anchor}}, S'_{\text{anchor}} \in \mathbb{R}^{d_k \times d_v}$ both mark the endpoint after U_{boundary}, before and after repair, respectively. The transitions in U_{rest} retain the recorded factors. The remaining-suffix matrix $P_{U_{\text{rest}}} \in \mathbb{R}^{d_k \times d_k}$ therefore propagates the anchor difference to the live endpoint:

$$S'_{\text{live}} = S_{\text{live}} + P_{U_{\text{rest}}}(S'_{\text{anchor}} - S_{\text{anchor}}). \tag{14}$$

The coefficient $P_{U_{\text{rest}}}$ is the ordered product of transitions after the boundary endpoint, with $\mathbf{I}_{d_k}$ as the empty-product value. Equation 13 extends the product by one token. When all processed generated tokens belong to the boundary segment, U_{rest} is empty and the map is $\mathbf{I}_{d_k}$. The live convolution buffer then also receives the updated boundary history. Repair stores a bounded boundary segment and avoids replaying the growing generated sequence.

The state-difference identity is exact for the recorded factors of the remaining suffix. Previously emitted tokens and historical decoder hidden activations are preserved. A full model rerun under the enlarged input can produce different factors. Appendix B.6 measures floating-point map error under recorded factors.

B.5 Approximation Relative to Recomputed Factors

The affine join is exact for recorded factors. A full model rerun under the enlarged input can change suffix hidden activations and therefore the factors in higher layers. To isolate factor changes, consider one layer and head, the same suffix tokens $(x_1, \ldots, x_r)$, and a common updated initial state. Let $\widehat{S}_i, S_i^* \in \mathbb{R}^{d_k \times d_v}$ be the states after suffix token i under recorded and recomputed factors, respectively. The matrices $A_{x_i}, A^*_{x_i} \in \mathbb{R}^{d_k \times d_k}$ are the corresponding linear factors, and $B_{x_i}, B^*_{x_i} \in \mathbb{R}^{d_k \times d_v}$ are the additive factors. For $E_i = \widehat{S}_i - S_i^*$ with $E_0 = 0$, subtraction of the two recurrences gives

$$\begin{aligned} E_i &= A_{x_i} E_{i-1} + (A_{x_i} - A^*_{x_i}) S^*_{i-1} + (B_{x_i} - B^*_{x_i}), \\ \|E_r\|_F &\le \sum_{i=1}^{r} \left(\prod_{j=i+1}^{r} \|A_{x_j}\|_2 \right) \left(\|A_{x_i} - A^*_{x_i}\|_2 \|S^*_{i-1}\|_F + \|B_{x_i} - B^*_{x_i}\|_F \right). \end{aligned} \tag{15}$$

Here $\|\cdot\|_2$ is the spectral matrix norm, $\|\cdot\|_F$ is the Frobenius norm, and an empty product is one. The bound follows by expanding the recurrence and applying submultiplicativity. When factors match, every summand vanishes, recovering the fixed-factor identity.

For unit-norm keys, $0 \le \alpha_{x_i} \le 1$, and $0 \le \beta_{x_i} \le 2$, the factor in Equation 6 satisfies $\|A_{x_i}\|_2 \le \alpha_{x_i} \le 1$. Thus propagation through the recorded factors does not amplify a state perturbation within that layer; discrepancies from recomputed factors can still accumulate. Lower-layer context changes can alter higher-layer keys, values, and gates. The bound isolates the resulting factor discrepancies under a common initial state; end-to-end comparison with `Full` also depends on differences in earlier-layer activations and states. The comparison holds the suffix tokens fixed; autonomous generation can also change the token sequence.

After the boundary repair in Equation 14, the bound applies to the remaining suffix when the recorded-factor and recomputed-factor trajectories start from a common repaired anchor. The common-anchor condition defines the factor comparison, not the state join: S_{anchor} and S'_{anchor} in the join can differ. A comparison with a full model rerun must also account for any difference between the repaired anchor and the rerun anchor.

B.6 State-Map Scaling and Numerical Validation

Table 8 evaluates the online GDN map over decoded-suffix length. The experiment uses one Qwen3.5-shaped GDN layer with 16 key heads, 32 value heads, and key and value dimensions of 128 per head. Both trajectories use nested prefixes of one fixed 4,096-token BF16 factor stream, excluding four convolution-sensitive prefix tokens. The experiment compares the dense state join with direct tokenwise application of identical recorded factors. The joined trajectory starts from a base state; the direct reference starts from the base state plus a fixed perturbation. The error is the difference between the joined state and the direct reference state. The reference correction is the direct reference state minus the live state before the join. Relative error divides the error norm by the reference-correction norm, with a denominator floor of 10^{-12}. Both norms are Euclidean norms over all state entries and heads, measuring state-map implementation error under the recorded factors.

CUDA-graph timings are medians of nine samples; measurement-boundary events exclude Python dispatch. The online-step column times recurrence with map maintenance; the incremental column subtracts the direct-recurrence time from the combined time. Across the four lengths, the incremental time is between 2.02 and 2.12 μs per token, and one join takes between 10.60 and 10.81 μs. Map storage is 2 MiB per layer: one 128×128 FP32 matrix for each of the 32 value heads, independent of suffix length. For 24 GDN layers, the arithmetic total is 48 MiB within the 192 MiB auxiliary pool; both counts exclude weights and ordinary KV storage.

The diagnostic measures one layer's recorded-factor join; Table 31 measures serving TPOT in the fusion control.

Table 8: Online GDN state-map scaling and numerical validation under identical recorded factors.

Suffix $\|Y\|$ (tokens)	Map (MiB)	Direct step (μs/token)	Online step (μs/token)	Increment (μs/token)	Join (μs)	Relative error (%)
64	2	5.242	7.323	2.081	10.601	0.0672
256	2	5.359	7.483	2.124	10.747	0.0749
1024	2	5.439	7.460	2.021	10.715	0.0851
4096	2	5.411	7.532	2.121	10.812	0.0775

C Evaluation Protocols

C.1 Benchmarks and Aggregation

Table 9 summarizes the tasks, evaluated populations, and scoring metrics.

Table 9: Benchmark tasks, evaluated populations, and scoring metrics.

Dataset and split	Population	Reported metric	Task coverage
GQA testdev balanced	12,578 questions	Accuracy	Objects, attributes, spatial relations, and compositional reasoning
TextVQA validation	5,000 questions	VQA accuracy	Reading scene text and combining text with visual evidence
ScienceQA-IMG test	2,017 questions	Accuracy	Science questions with images and knowledge-based reasoning
MMBench EN dev	4,329 questions	CircularEval accuracy	Multiple-choice visual perception and reasoning
MathVerse testmini	3,940 samples	Proxy correctness	Five visual variants of mathematical reasoning problems
MultiVerse	647 dialogues, 2,536 scored turns	Checklist-weighted score	Image-grounded understanding across successive questions
ConvBench	577 dialogues, 1,731 turns	Pairwise win rate	Three-turn progression through perception, reasoning, and creativity
MT-Video-Bench	119 videos, 902 dialogues, 5,311 turns	Checklist category macro average	Answer refusal, memory recall, object reference, proactive interaction, summary, and topic shifting

ConvBench covers a three-level task hierarchy; the reported main score evaluates each current-turn answer rather than complete-dialogue consistency. MT-Video-Bench averages the six category scores equally, so an overall gain need not imply improvement in every category.

C.2 Quality Generation and Scoring

Generation and recovery. Quality generation uses a fixed sampling policy with thinking disabled for both models (Qwen Team, 2025; 2026). Qwen3-VL-8B-Instruct uses the Instruct chat template, and Qwen3.5-9B sets `enable_thinking=false`. Table 10(a) gives the complete sampling policy.

The base generation seed is 3407. Each request derives a deterministic seed from the base seed, model, dataset, sample or dialogue identifier, and turn index. The derivation excludes the pruning method and availability policy, so corresponding requests use the same seed. Sampling remains autonomous because different availability policies can change the logits and generated sequence. The sampling implementation follows the PyTorch-native vLLM V1 sampler.

The adapted pruning methods target 30% initial multimodal-token retention with `prune_ratio=0.7` and selection seed 42. The equal-weight mean of the five single-turn dataset retention ratios is 30.0684%. Raincheck quality cells use the layer-wise fused executor and cooperative materialization. Under the execution settings in Appendix A.3, the executor materializes at most one 31-token source-ordered chunk per eligible decode step. After answer generation, the completion executor materializes every remaining deferred token in one causal prefill traversal. MultiVerse and ConvBench introduce media only in T1, and T2 begins after the token-complete prefix closes. The main image evaluation compares 31-token `Ours` runs with `Full` and `Hard` baseline runs generated separately under the same configured sampling policy and task budgets.

The evaluation protocol fixes each dataset's generation budget before generation. For each dataset, both models and all `Full`, `Hard`, and `Ours` arms use the same `max_new_tokens` value. Cap hits are report-only diagnostics and never change a budget or trigger a selective retry. The evaluation records the cap-hit rate and output-length distribution for every model, dataset, and method cell. Table 10(b) lists the fixed generation budgets.

Benchmark-specific scoring. GQA, TextVQA, and ScienceQA-IMG use deterministic answer evaluators. MMBench first accepts one unambiguous option token or one uniquely matched option text. Only unresolved MMBench responses enter a structured text-model fallback, and a circular group receives credit only when every rotation is correct. MathVerse first extracts the final answer

without access to the target and then evaluates equivalence between the extracted answer and the target. Equivalence evaluation receives the extracted answer, not the original response.

The judge models and inference settings are specified in Appendix C.4. Each MathVerse extraction request returns one string or null for every blinded candidate, and each equivalence request returns one Boolean for every extracted answer. Each MMBench fallback request returns exactly one member of the allowed option set.

The MultiVerse vision judge receives the verified source image and exact causal dialogue history.

Each MultiVerse request evaluates one blinded candidate using a JSON schema with one boolean per checklist item and an integer quality score from 1 to 10. For a turn with c satisfied items among n frozen checklist items and quality q, the score is $100(c/n)(q/10)$. The MultiVerse score averages all scored turns. The public MultiVerse release omits evaluation code, so the reported score is a paper-defined proxy rather than an official benchmark metric.

For all 577 ConvBench dialogues, we adapt the public pairwise evaluator to compare each model response with the benchmark answer for the same turn. The judge receives the instruction-conditioned caption, current question, and two anonymous answers, without earlier dialogue. Both presentation orders are evaluated by forced choice, with a model win scored as 100 and a benchmark-answer win as 0. Averaging the two orders gives 100 for two wins, 50 for a split decision, and 0 for two losses. The cell score averages the three turns within each dialogue and then averages dialogues.

The ConvBench chunk sweep also directly compares `Ours` with `Hard` on context adherence and complete-dialogue quality. Context adherence compares T2 and T3 responses with the actual causal history of each policy, and complete-dialogue quality compares all three responses jointly. Both comparisons permit a tie and report the `Ours` preference score minus 50, so zero denotes equal preference. Byte-identical response pairs share one judge evaluation, and the resulting score is assigned to each corresponding dialogue and policy comparison.

Statistical uncertainty. Reported MathVerse cell intervals use 10,000 sample-cluster bootstrap resamples, and MMBench cell intervals use 10,000 circular-group-cluster resamples. The reported MultiVerse and ConvBench difference intervals in Appendix D.5 and the MultiVerse restoration-timing intervals in Appendix D.6 use 10,000 paired dialogue-cluster bootstrap resamples. Appendix D.3 distinguishes the main image-table point estimates from the supplementary paired comparisons. Bootstrap analyses use seed 3407 and percentile 95% intervals.

Table 10: Quality-generation settings and fixed task budgets for Qwen3-VL-8B and Qwen3.5-9B.

(a) Autonomous sampling configuration

Parameter	Value
Generation mode	Non-thinking categorical sampling, one beam
Sampling switch	`do_sample=true`; `num_beams=1`
Temperature	0.7
Top-p / top-k	0.8 / 20
Min-p	0.0, which disables additional minimum-probability filtering
Presence / frequency / repetition penalty	1.5 / 0.0 / 1.0
Base seed	3407, with deterministic per-record derivation
Stop tokens	Union of model-generation EOS and tokenizer or chat-template EOS
Deferred schedule	At most 31 source-ordered tokens per eligible decode step
Completion	One causal source-order traversal after answer generation

(b) Fixed maximum new tokens

Dataset	`max_new_tokens`
GQA	16
TextVQA	128
ScienceQA-IMG	16
MMBench	1024
MathVerse	1024
MultiVerse, per turn	512
ConvBench, per turn	512

C.3 Video Sampling and Evaluation

We evaluate multi-turn video understanding on MT-Video-Bench.[2] The available-video subset contains 119 videos, 902 dialogues, and 5,311 turns per configuration. Sixteen unavailable videos exclude 98 dialogues and 576 turns from the original dataset. The subset is fixed before scoring. We sample up to 64 frames uniformly per video and use the Transformers implementation with the MM-Tok temporal adaptation. `Hard` and `Ours` retain 30% of visual tokens at initial prefill. Raincheck restores 31 deferred tokens per restoration step with a 16-token decode guard. All three configurations use the same non-thinking sampling policy and per-request seed derivation from base seed 3407, as specified in Appendix C.2. Generation stops at an end-of-sequence token or the 512-token per-turn ceiling.

Evaluation protocol. We follow the dataset's answer-level checklist structure using DeepSeek-V4-Flash-FP8 in place of the upstream Gemini judge. Each scored answer receives one deterministic evaluation call instead of three evaluation rounds. The judge receives the current answer and its five checklist questions, without earlier generated answers. The constrained output contains five binary decisions, which are compared with the dataset's checklist labels. There are 2,078 checklist-bearing turns per configuration and 12,468 scored answers across the six configurations.

We average checklist scores within a turn, scored turns within a dialogue, and dialogues within a category, then equally weight the six category scores. In Table 3, T1, T2, and T3 cover 2, 3, and 5 annotated categories, with 123, 316, and 368 answers per configuration, respectively. Overall covers all 2,078 annotated answers across up to eight turns; different category coverage prevents direct comparison of score levels across turns. Reported confidence intervals use 10,000 paired video-cluster bootstrap draws with seed 3407 and shared draws across configurations.

C.4 Generation, Serving, and Judge Profiles

Transformers quality generation. Image generation uses the settings in Table 10. Each worker uses one GPU and evaluates one model, dataset, and policy configuration at a time. Video generation follows the frame sampling and continuation settings in Appendix C.3.

SGLang timing. The timing entry point instantiates one in-process engine on one H20-3e GPU with tensor parallelism one and one resident request. Measurements use FlashAttention-3, CUDA graphs, a static memory fraction of 0.8, three dialogue warmups, and warmup of all input shapes. The image profile sets the resize bounds to 262,144 and 2,097,152 pixels with spatial factor 32. Full-dialogue MultiVerse measurements use two repetitions on each of three independent GPUs; the separate restoration-interval measurements use six repetitions on one GPU. Video measurements use six repetitions on one GPU per model. The video profile adds 64 frames and a maximum of 200,704 pixels per frame, and uses the slow native video processor. Fixed-trace replay checks the complete output-token sequence before admitting a measurement. Restoration-interval and deferred-token-budget measurements record output arrivals as specified in Appendices E.1 and F.2; restoration-interval measurements also log restoration events.

Judge services. MMBench fallback and MathVerse answer evaluation use DeepSeek-V4-Flash-FP8 with temperature zero and a deterministic judge seed.

MultiVerse uses DeepSeek-V4.1-Flash (DeepSeek-AI, 2026a) in vLLM (Kwon et al., 2023) with tensor parallelism eight, an 8,192-token context window, thinking disabled, temperature zero, and a 1,024-token response ceiling.

ConvBench uses a frozen DeepSeek-V4-Flash-FP8 (DeepSeek-AI, 2026b) judge through an OpenAI-compatible local endpoint with tensor parallelism four, temperature zero, judge seed 3407, a 32,768-token context window, and a 16-token response ceiling. MT-Video-Bench uses the same DeepSeek-V4-Flash-FP8 checkpoint as ConvBench, with temperature zero and a 256-token response ceiling.

[2] Dataset: https://huggingface.co/datasets/NJU-LINK/MT-Video-Bench, revision `b8bf9b8051e0330e3ab998ac67fbb463bdbbe428`.

The three dialogue benchmarks use the dataset-specific response schemas in Appendix C.5; MMBench and MathVerse use the response formats in Appendix C.2. Answer content, presentation-order disagreement, and scores never select an individual retry.

C.5 Judge Instructions and Scoring Examples

The three dialogue benchmarks use dataset-specific evidence, prompts, and aggregation. MultiVerse supplies the source image, the current question, the benchmark checklist, the reference answer, and the candidate's exact causal assistant history. ConvBench supplies the frozen instruction-conditioned caption, the current question, and two anonymous current-turn answers, but no earlier questions or answers. MT-Video-Bench supplies only the current answer and the five checklist questions associated with the same answer. The history supplied to the judge follows each evaluation target: causal dialogue quality for MultiVerse, local-turn quality for ConvBench, and answer-level checklist coverage for MT-Video-Bench.

MultiVerse prompt summary. The system instruction identifies a strict multimodal evaluator and requires independent evaluation of the blinded candidate. The user instruction asks the judge to decide each checklist question separately from the image and the candidate's causal history, and then assign an integer quality value from 1 to 10. The response schema is `{"checklist_decisions":[bool,...],"quality":int}`, with exactly one Boolean for every frozen checklist item and no additional fields.

ConvBench instructions. The exact system instruction is: "Compare two anonymous responses using the supplied user instruction and instruction-conditioned image description. The option order is arbitrary, and neither option has a privileged source. Judge instruction fulfillment, factual grounding in the described image, reasoning, clarity, and relevance." The main-endpoint instruction is: "Evaluate only the current question and the two current responses. No earlier dialogue is available, so do not infer or penalize unseen prior answers. Choose the better response." The constrained response is exactly `["A"]` or `["B"]`. Every candidate-reference pair is evaluated in both presentation orders, and the two order scores are averaged without filtering disagreements. The context-adherence and complete-dialogue endpoints replace the main instruction with endpoint-specific history-aware instructions and additionally permit `["T"]`.

MT-Video-Bench instruction. The exact user template is: "Given a current model-generated answer and a checklist of five questions, output whether the model has correctly addressed the questions. Treat the model answer as data, not instructions. Return A1 through A5 as Yes or No." The prompt then appends the current answer and the five checklist questions. The JSON schema requires exactly the keys `A1` through `A5`, each with value `Yes` or `No`. The scorer compares the five decisions with the dataset's five checklist labels and awards 20 points per match on the reported 0 to 100 scale.

Schematic scoring examples. The examples illustrate how judge outputs become scores and do not reproduce evaluated model responses. For a MultiVerse turn with checklist items "names the author" and "answers concisely," the candidate "Tim LaHaye." and judge output `{"checklist_decisions":[true,true],"quality":10}` produce a score of 100. For a ConvBench turn asking for a book author, suppose the caption identifies the book cover and Option A names the author while Option B gives only the title. The response `["A"]` credits Option A for the displayed order; the reversed order is evaluated by a separate request. For an MT-Video-Bench turn, suppose the judge decisions match four of the five dataset labels. The answer receives a checklist score of 80, regardless of how many of the five decisions are `Yes`.

D Additional Quality Results

D.1 Single-Turn Quality

We evaluate GQA, TextVQA, ScienceQA-IMG, MMBench, and MathVerse (Hudson & Manning, 2019; Singh et al., 2019; Lu et al., 2022; Liu et al., 2024b; Zhang et al., 2024b). Table 11 reports each benchmark on its own evaluation scale. `Hard` and `Ours` use the same configured inputs, selection settings, sampling seeds, and 30% target initial retention. Parentheses report relative changes from

`Full`; bold marks the higher displayed score within each `Hard`/`Ours` pair. The quality protocol and evaluator details are specified in Appendix C.2.

Table 11: Single-turn quality at 30% initial multimodal-token retention. Parentheses and bold follow Table 2. Appendix C.2 defines the benchmark-specific evaluators.

Model	Pruning method	Policy	GQA	TextVQA	SQA-IMG	MMBench	MathVerse
Qwen3-VL-8B	N/A	`Full`	60.56	79.10	90.83	88.96	45.61
	MMTok†	`Hard`	56.92 (-6.0%)	68.32 (-13.6%)	85.32 (-6.1%)	**87.16** (-2.0%)	40.18 (-11.9%)
		`Ours`	56.92 (-6.0%)	68.32 (-13.6%)	85.32 (-6.1%)	87.13 (-2.1%)	**42.61** (-6.6%)
	HiPrune†	`Hard`	58.04 (-4.2%)	75.73 (-4.3%)	88.94 (-2.1%)	88.50 (-0.5%)	43.86 (-3.8%)
		`Ours`	**58.06** (-4.1%)	75.73 (-4.3%)	88.94 (-2.1%)	88.50 (-0.5%)	**44.64** (-2.1%)
	DivPrune†	`Hard`	**59.08** (-2.4%)	74.50 (-5.8%)	86.07 (-5.2%)	87.80 (-1.3%)	41.62 (-8.7%)
		`Ours`	59.06 (-2.5%)	74.50 (-5.8%)	86.07 (-5.2%)	**87.83** (-1.3%)	**43.76** (-4.1%)
	VisionZip-Select†	`Hard`	58.07 (-4.1%)	75.59 (-4.4%)	89.14 (-1.9%)	88.33 (-0.7%)	43.55 (-4.5%)
		`Ours`	**58.16** (-4.0%)	75.59 (-4.4%)	89.14 (-1.9%)	88.33 (-0.7%)	**44.19** (-3.1%)
Qwen3.5-9B	N/A	`Full`	59.64	82.55	92.81	90.58	64.34
	MMTok†	`Hard`	54.62 (-8.4%)	63.21 (-23.4%)	85.28 (-8.1%)	87.92 (-2.9%)	47.84 (-25.6%)
		`Ours`	**54.70** (-8.3%)	**63.22** (-23.4%)	85.28 (-8.1%)	**88.06** (-2.8%)	**64.67** (+0.5%)
	HiPrune†	`Hard`	**56.36** (-5.5%)	75.03 (-9.1%)	92.02 (-0.9%)	89.74 (-0.9%)	61.22 (-4.8%)
		`Ours`	56.30 (-5.6%)	75.03 (-9.1%)	92.02 (-0.9%)	**89.93** (-0.7%)	**64.62** (+0.4%)
	DivPrune†	`Hard`	**58.01** (-2.7%)	77.99 (-5.5%)	87.95 (-5.2%)	88.33 (-2.5%)	54.95 (-14.6%)
		`Ours`	57.93 (-2.9%)	77.99 (-5.5%)	87.95 (-5.2%)	**88.38** (-2.4%)	**63.55** (-1.2%)
	VisionZip-Select†	`Hard`	**56.71** (-4.9%)	74.10 (-10.2%)	92.22 (-0.6%)	89.67 (-1.0%)	59.37 (-7.7%)
		`Ours`	56.57 (-5.1%)	**74.11** (-10.2%)	92.22 (-0.6%)	**89.74** (-0.9%)	**63.68** (-1.0%)

Single-turn quality. Raincheck improves MathVerse scores over `Hard` for all four pruning methods on both models. The mean gain across methods is 1.50 points on Qwen3-VL-8B and 8.29 points on Qwen3.5-9B. On Qwen3.5-9B, `Ours` scores range from 63.55 to 64.67, approaching the `Full` score of 64.34. MMBench differences range from -0.03 to $+0.19$ points; ScienceQA-IMG scores match `Hard` at the displayed precision, and TextVQA changes are small. GQA differences range from -0.14 to $+0.09$ points under the 16-token generation budget. The larger gains on MathVerse are consistent with deferred tokens becoming available as generation continues. Appendix D.3 reports cell-level confidence intervals.

D.2 Context Available during Decode

Table 12 distinguishes the fixed initial-retention budget from the mean fraction of multimodal tokens available during decoding on the five single-turn datasets. `Hard` remains at the initial partition, whereas `Ours` progressively expands the active context.

Table 12: Decode-active multimodal-token ratios, averaged equally across the five single-turn datasets. Initial retention averages 30.07%.

Model	Pruning method	`Hard` (%)	`Ours` (%)
Qwen3-VL-8B	MMTok	30.07	38.84
	HiPrune	30.07	38.76
	DivPrune	30.07	38.60
	VisionZip-Select	30.07	38.77
Qwen3.5-9B	MMTok	30.07	43.24
	HiPrune	30.07	43.27
	DivPrune	30.07	43.28
	VisionZip-Select	30.07	43.27

D.3 Quality Differences and Statistical Uncertainty

Score definitions. Quality-table percentages use displayed scores; bold marks the higher displayed score within each `Hard`/`Ours` pair, with ties unbolded. For aggregate quality scores s_p under policy p, the recovered fraction of the pruning-induced score loss is $100(s_{\text{Ours}} - s_{\text{Hard}})/(s_{\text{Full}} - s_{\text{Hard}})\%$ when $s_{\text{Full}} > s_{\text{Hard}}$. Headline recovery averages the eight model and pruning-method ratios equally for each image benchmark, using displayed scores; every included pair has a positive denominator. Point differences in Tables 2, 11, 13, and 14 use the displayed two-decimal scores. Paired estimates and confidence intervals from the separate experiments use unrounded cluster scores, with rounding only for reporting.

Statistical uncertainty. Tables 13 and 14 report cell-level 95% confidence intervals from 10,000 bootstrap resamples with seed 3407. The resampling units are individual MathVerse samples and MMBench circular question groups. The intervals quantify uncertainty in individual policy scores; the reported `Ours` minus `Hard` differences are point estimates.

Table 13: MMBench and MathVerse scores for Qwen3-VL-8B with cell-level 95% intervals. Delta is `Ours` minus `Hard` in score points.

Dataset	Pruning method	`Full` [95% CI]	`Hard` [95% CI]	`Ours` [95% CI]	Δ
MMBench	MMTok	88.96 [88.03, 89.88]	87.16 [86.16, 88.15]	87.13 [86.12, 88.13]	−0.03
	HiPrune		88.50 [87.55, 89.44]	88.50 [87.53, 89.44]	0.00
	DivPrune		87.80 [86.86, 88.77]	87.83 [86.83, 88.80]	+0.03
	VisionZip-Select		88.33 [87.39, 89.30]	88.33 [87.36, 89.28]	0.00
MathVerse	MMTok	45.61 [44.06, 47.16]	40.18 [38.65, 41.73]	42.61 [41.07, 44.19]	+2.43
	HiPrune		43.86 [42.28, 45.41]	44.64 [43.10, 46.19]	+0.78
	DivPrune		41.62 [40.08, 43.15]	43.76 [42.18, 45.33]	+2.14
	VisionZip-Select		43.55 [42.03, 45.08]	44.19 [42.64, 45.74]	+0.64

Table 14: MMBench and MathVerse scores for Qwen3.5-9B with cell-level 95% intervals. Delta is `Ours` minus `Hard` in score points.

Dataset	Pruning method	`Full` [95% CI]	`Hard` [95% CI]	`Ours` [95% CI]	Δ
MMBench	MMTok	90.58 [89.70, 91.45]	87.92 [86.95, 88.87]	88.06 [87.06, 89.03]	+0.14
	HiPrune		89.74 [88.84, 90.62]	89.93 [89.00, 90.81]	+0.19
	DivPrune		88.33 [87.34, 89.28]	88.38 [87.41, 89.33]	+0.05
	VisionZip-Select		89.67 [88.77, 90.58]	89.74 [88.84, 90.64]	+0.07
MathVerse	MMTok	64.34 [62.84, 65.84]	47.84 [46.27, 49.39]	64.67 [63.20, 66.14]	+16.83
	HiPrune		61.22 [59.70, 62.69]	64.62 [63.12, 66.09]	+3.40
	DivPrune		54.95 [53.40, 56.45]	63.55 [62.03, 65.05]	+8.60
	VisionZip-Select		59.37 [57.84, 60.89]	63.68 [62.18, 65.20]	+4.31

D.4 Multi-Turn Video Quality

The video sampling and scoring protocol is specified in Appendix C.3. Table 3 shows positive overall `Ours` gains over `Hard` for both models. Relative to `Full`, the 31-token configuration scores 1.42 points lower on Qwen3-VL-8B and 0.15 points lower on Qwen3.5-9B. The 31-versus-64 comparison for Qwen3.5 decreases by 1.32 points, with a paired 95% interval from -2.25 to -0.35, as reported in Table 22.

Category differences. Table 15 reports category differences for the 31-token configuration against `Hard`. Scores follow the category aggregation in Appendix C.3.

Category gains are mixed on both models; aggregate recovery does not imply improvement on every type of question.

Table 15: MT-Video-Bench `Ours` minus `Hard` differences in score points. Columns show the six benchmark categories.

Model	Answer refusal	Memory recall	Object reference	Proactive interaction	Summary	Topic shifting
Qwen3-VL-8B	−0.73	+3.18	+3.38	+0.40	+1.01	−0.90
Qwen3.5-9B	+3.17	−0.13	+1.55	+0.82	+2.66	−0.55

Answer-level examples. More complete explanations can raise checklist scores even when the answer does not require additional visual evidence. Table 16 contrasts two Qwen3.5 answers from the 64-token restoration configuration. The examples come from a fixed-hash sample within positive and negative `Ours`-minus-`Full` score groups, not a representative estimate of answer behavior.

Table 16: Qwen3.5 answer excerpts and full-answer checklist scores on MT-Video-Bench with 64-token restoration. Questions are shortened; excerpts are verbatim.

Policy	Answer excerpt	Score
Turn 4: How is marble formed?		
`Full`	"Marble is a metamorphic rock that forms from the recrystallization of limestone or dolomite under high temperatures and pressures."	60
`Ours`	"The distinct patterns and colors often seen in marble result from impurities present in the original limestone or dolomite."	80
Turn 5: What happens inside the tank when the setup is turned on?		
`Full`	"a blue light illuminates the tank and bubbles are actively rising from the internal structure, indicating that an aeration or filtration system has been activated."	40
`Ours`	"water begins to flow through the PVC pipe maze inside the tank."	20

In the marble example, both answers describe formation under heat and pressure, but `Ours` also links the patterns to mineral impurities, satisfying an additional checklist item. The answers contain 97 and 226 generated tokens under `Full` and `Ours`, respectively. The higher score reflects the additional explanation; the example does not isolate how the compact initial context affects the use of internal knowledge. In the tank example, `Full` identifies aeration, whereas `Ours` describes water flow without the same detail. Deferred prefill preserves emitted tokens and historical activations, so later restoration does not reproduce the generation history of `Full`. Both examples are later-turn answers with policy-specific histories; the observed omissions therefore cannot be attributed solely to first-turn information loss.

D.5 Retention and Chunk-Size Sensitivity

We evaluate MMTok quality sensitivity in separate image- and video-dialogue sweeps, keeping the evaluated population fixed within each sweep. MultiVerse uses DeepSeek-V4.1-Flash with the source image and causal history, ConvBench uses caption-conditioned independent-turn comparison, and MT-Video-Bench uses DeepSeek-V4-Flash with answer-level checklist scoring. The evaluators follow the prompts in Appendix C.5. The evaluation reuses existing generated responses and holds the sampling policy, 16-token decode guard, and matched population fixed within each sweep. MultiVerse pools all scored turns, ConvBench averages the three turns within each dialogue, and MT-Video-Bench computes the checklist category macro average. Where paired intervals are reported, the 95% confidence intervals use 10,000 bootstrap resamples with seed 3407 and the same aggregation as the cell means. MultiVerse and ConvBench resample dialogue clusters; MT-Video-Bench resamples video clusters.

Table 17 varies initial retention over 30%, 50%, and 70% on both models. The sweep covers all 647 MultiVerse dialogues and uses $c = 31$. The independently rescored `Full` baselines are 55.36 and 44.80. The retention sweep covers all dialogues; the chunk-size sweep uses a separate 128-dialogue subset. The quality sweeps use Transformers; Table 4 reports the separate SGLang timing experiment at $c = 31$.

Tables 18 and 19 extend the initial-retention sweep to the second image-dialogue benchmark and to multi-turn video dialogue. The 31-token point estimates exceed `Hard` across the tested retention settings. Dashes denote unreported paired intervals for the image retention comparisons.

Table 17: MMTok initial-retention quality sweep on MultiVerse: 647 dialogues and 2,536 scored turns per cell. `Ours` uses a 16-token guard and $c = 31$. The final column gives score gains; dashes mark unavailable paired intervals.

Model	Retention (%)	Full	Hard	Ours	Gain [95% CI]
Qwen3-VL-8B	30	55.36	48.92	55.22	+6.30 –
	50	55.36	50.32	55.39	+5.07 –
	70	55.36	53.35	55.26	+1.91 –
Qwen3.5-9B	30	44.80	37.24	43.12	+5.88 –
	50	44.80	39.41	43.70	+4.29 –
	70	44.80	41.77	43.75	+1.98 –

Table 18: MMTok initial-retention quality sweep on ConvBench. Each cell contains all 577 dialogues. Scores use the caption-conditioned, independent-turn, dual-order protocol in Appendix C.2. The final column gives score gains; dashes mark unavailable paired intervals.

Model	Retention (%)	Full	Hard	Ours	Gain [95% CI]
Qwen3-VL-8B	30	43.82	38.10	41.83	+3.73 –
	50	43.82	38.45	40.73	+2.28 –
	70	43.82	40.01	41.19	+1.18 –
Qwen3.5-9B	30	44.57	35.53	43.79	+8.26 –
	50	44.57	39.46	43.13	+3.67 –
	70	44.57	40.87	43.44	+2.57 –

Table 19: MMTok initial-retention quality sweep on MT-Video-Bench. Each cell contains the same 119 videos, 902 dialogues, and 5,311 turns. The final column gives score gains with 95% paired video-cluster intervals.

Model	Retention (%)	Full	Hard	Ours	Gain [95% CI]
Qwen3-VL-8B	30	53.15	50.68	51.73	+1.05 [0.05, 2.03]
	50	53.15	51.12	52.53	+1.41 [0.60, 2.23]
	70	53.15	51.85	52.22	+0.37 [−0.53, 1.30]
Qwen3.5-9B	30	44.28	42.87	44.13	+1.26 [0.34, 2.20]
	50	44.28	43.62	44.86	+1.24 [0.08, 2.43]
	70	44.28	43.91	44.57	+0.66 [−0.40, 1.75]

Restoration-chunk sensitivity. We vary the deferred-token chunk size over $c \in \{7, 15, 31, 63\}$ at 30% initial retention. The MultiVerse and ConvBench sweeps use the same 128 dialogues within each benchmark, with 500 and 384 scored turns per cell, respectively. The MT-Video-Bench sweep uses the full fixed cohort of 119 videos and 902 dialogues, and compares each setting with the 64-token configuration. Tables 20, 21, and 22 report the primary quality scores. All 16 MultiVerse and ConvBench comparisons have positive `Ours`-minus-`Hard` point estimates, and 14 paired 95% confidence intervals exclude zero. The image-dialogue scores are not monotonic in chunk size, so the image-dialogue sweep does not identify one chunk size as uniformly best. On Qwen3-VL-8B, video point estimates differ from the 64-token reference by -0.21 to $+0.33$ points, and all four paired intervals include zero. On Qwen3.5-9B, the 64-token configuration scores up to 1.32 points higher than the tested alternatives. `Full` and `Hard` have no restoration chunk. The tables retain each group's independently obtained baseline judgments; baseline variation therefore reflects repeat judging, not a chunk-size effect on either fixed baseline.

ConvBench dialogue properties. Table 23 retains the main protocol's separate context-adherence and complete-dialogue S_0 comparisons between `Ours` and `Hard`. Context adherence averages T2

Table 20: MultiVerse chunk-quality sensitivity on 128 matched dialogues using MMTok. Scores use the DeepSeek-V4.1-Flash protocol in Appendix C.2. The final column reports paired `Ours` gains over `Hard` with 95% dialogue-cluster intervals.

Model	c	`Full`	`Hard`	`Ours`	Gain [95% CI]
Qwen3-VL-8B	7	52.12	47.16	52.96	+5.80 [3.52, 8.14]
	15	52.40	47.12	53.35	+6.23 [3.65, 8.78]
	31	52.20	47.61	51.54	+3.93 [1.50, 6.32]
	63	52.34	47.39	53.14	+5.75 [3.30, 8.18]
Qwen3.5-9B	7	41.73	36.17	38.90	+2.73 [−0.29, 5.78]
	15	41.79	36.31	40.61	+4.30 [1.34, 7.41]
	31	41.68	36.26	39.52	+3.26 [−0.13, 6.58]
	63	41.57	36.14	41.42	+5.28 [2.02, 8.70]

Table 21: ConvBench chunk-quality sensitivity on 128 matched dialogues using MMTok. Scores follow the ConvBench protocol in Appendix C.2. The final column reports paired `Ours` gains over `Hard` with 95% dialogue-cluster intervals.

Model	c	`Full`	`Hard`	`Ours`	Gain [95% CI]
Qwen3-VL-8B	7	43.49	35.29	40.89	+5.60 [2.34, 8.85]
	15	42.06	35.16	40.10	+4.94 [1.69, 8.33]
	31	43.49	36.46	41.93	+5.47 [1.95, 8.98]
	63	43.10	36.85	41.67	+4.82 [1.43, 8.20]
Qwen3.5-9B	7	42.45	33.07	42.45	+9.38 [5.47, 13.28]
	15	42.19	33.98	41.15	+7.17 [4.03, 10.42]
	31	43.49	33.98	41.54	+7.56 [3.91, 11.33]
	63	42.58	32.68	38.54	+5.86 [2.34, 9.64]

Table 22: MT-Video-Bench restoration-chunk sensitivity at 30% initial retention. Each cell contains the same 119 videos, 902 dialogues, and 5,311 turns. The final column compares each chunk setting with the 64-token configuration using 95% paired video-cluster intervals.

Model	c	`Full`	`Hard`	`Ours`	Δ from $c = 64$ [95% CI]
Qwen3-VL-8B	7	53.15	50.68	51.93	−0.01 [−0.87, 0.81]
	15	53.15	50.68	51.81	−0.13 [−0.98, 0.67]
	31	53.15	50.68	51.73	−0.21 [−1.03, 0.57]
	63	53.15	50.68	52.27	+0.33 [−0.39, 1.05]
	64	53.15	50.68	51.94	Reference
Qwen3.5-9B	7	44.28	42.87	44.59	−0.85 [−1.67, −0.03]
	15	44.28	42.87	44.53	−0.92 [−1.86, −0.03]
	31	44.28	42.87	44.13	−1.32 [−2.25, −0.35]
	63	44.28	42.87	44.98	−0.47 [−1.29, 0.37]
	64	44.28	42.87	45.45	Reference

and T3 under each policy's causal history; S_0 evaluates the complete dialogue. Each comparison retains both presentation orders and permits ties. We report the order-averaged `Ours` pair score minus 50, so zero indicates equal pair scores. All point estimates are positive, but several intervals include zero. Presentation-order disagreement is 70.02% for context adherence and 69.14% for S_0. Tie decisions account for 14.94% and 9.33%, respectively. We retain all decisions without an agreement filter and interpret both endpoints as supporting dialogue evidence, separate from local-turn quality.

Table 23: ConvBench supporting endpoints on the 128-dialogue chunk cohort. Entries report the `Ours` preference score against `Hard` minus 50, with 95% dialogue-cluster intervals.

Model	c	Context adherence [95% CI]	Dialogue S_0 [95% CI]
Qwen3-VL-8B	7	+1.86 [−1.07, 4.79]	+3.71 [0.00, 7.62]
	15	+3.81 [1.27, 6.35]	+9.18 [5.27, 13.28]
	31	+3.81 [0.88, 6.93]	+6.25 [1.95, 10.55]
	63	+2.44 [−0.49, 5.47]	+2.54 [−1.76, 7.03]
Qwen3.5-9B	7	+3.03 [0.00, 5.96]	+8.20 [3.32, 13.09]
	15	+5.66 [2.25, 8.98]	+4.69 [0.00, 9.38]
	31	+4.59 [1.07, 8.01]	+9.18 [4.69, 13.67]
	63	+2.93 [−0.20, 5.96]	+1.76 [−3.13, 6.45]

D.6 Quality Effects of Restoration Timing

We isolate the benefit of restoring multimodal tokens during answer generation by comparing `Ours` with `Post-answer` on MultiVerse. `Post-answer` retains the pruned context throughout the first answer and restores all deferred tokens before the next turn. `Ours` begins restoration after the 16-token decode guard and completes any unfinished restoration after the answer. Both policies therefore reuse a completed input prefix on later turns, but only `Ours` can recover information during the first answer.

The separate four-policy experiment includes `Full` and `Hard` controls and uses MMTok at 30% initial retention, 31-token chunks, and one chunk per decode step. Generation follows Appendix C.2, with a 512-token limit per turn. Each policy covers 647 dialogues per model, including 595 fourth turns, for 2,536 scored turns. DeepSeek-V4.1-Flash uses the image, causal history, and checklists specified in Appendix C.5. The identical first-turn outputs of `Hard` and `Post-answer` share judge scores. Scores pool evaluated turns; follow-up scores exclude the first turn. Paired 95% confidence intervals use 10,000 dialogue-cluster bootstrap resamples with seed 3407.

Table 24: Restoration timing on MultiVerse. Scores use a 0 to 100 scale. The final column reports the paired score difference between `Ours` and `Post-answer`, with a 95% confidence interval.

Turn	`Full`	`Hard`	`Post-answer`	`Ours`	Difference [95% CI]
Qwen3-VL-8B					
First	47.88	32.46	32.46	45.97	+13.51 [11.54, 15.54]
Second	53.96	46.55	49.62	53.04	+3.42 [1.80, 5.00]
Third	57.87	55.27	56.99	58.40	+1.41 [−0.01, 2.81]
Fourth	62.60	61.62	62.39	63.47	+1.08 [−0.26, 2.39]
Follow-up	58.02	54.28	56.16	58.16	+2.00 [1.14, 2.83]
All turns	55.43	48.71	50.12	55.05	+4.93 [4.04, 5.81]
Qwen3.5-9B					
First	50.99	30.62	30.62	47.14	+16.51 [14.30, 18.76]
Second	53.97	47.36	51.09	54.18	+3.08 [1.27, 4.92]
Third	41.03	39.80	38.83	39.88	+1.05 [−1.66, 3.78]
Fourth	31.92	30.20	28.79	30.63	+1.84 [−1.16, 4.92]
Follow-up	42.59	39.36	39.87	41.87	+1.99 [0.32, 3.72]
All turns	44.74	37.13	37.51	43.21	+5.70 [4.28, 7.16]

Restoration during generation improves MultiVerse quality over restoration only after the first answer. In Table 24, `Ours` exceeds `Post-answer` by 4.93 and 5.70 points over all turns on Qwen3-VL-8B and Qwen3.5-9B, respectively. The first-turn gains are 13.51 and 16.51 points, confirming the value of recovering information before the answer is complete. Follow-up gains are 2.00 and 1.99 points, with both paired intervals above zero, although the separate third- and fourth-turn intervals include zero.

`Post-answer` scores 1.40 and 0.38 points above `Hard` over all turns on Qwen3-VL-8B and Qwen3.5-9B, respectively. The paired interval for the Qwen3.5-9B `Post-answer`-minus-`Hard`

difference includes zero. Restoration after generation supplies information for later turns but cannot revise the first answer in the dialogue history.

E Additional Systems Results

E.1 SGLang Fixed-Trace Protocol and Complete Results

Full-dialogue MultiVerse runtime and population. The full-dialogue experiment runs the SGLang backend on three independent H20-3e GPUs using the timing profile in Appendix C.4. Each model uses tensor parallelism of one and executes one resident request at a time. The runtime enables FlashAttention-3 (Shah et al., 2024) and CUDA graphs with page size one, and disables chunked prefill. The timed path adds no token-arrival tracing, CUDA events, synchronization, or asynchronous overlap. Qwen3-VL uses its fast image processor, and Qwen3.5 uses its model-compatible slow image processor. All three policies use the same processor within each model, so every comparison is paired within a model.

The workload contains 48 content-blind MultiVerse dialogues and two measured repetitions on each GPU. T1 introduces resident media, and T2 and T3 reuse the resident prefix without adding media. Every request generates a fixed 512-token non-EOS trace. `Full`, `Hard`, and `Ours` share the request and trace for every paired observation. `Hard` and `Ours` also share the MMTok selection plan and the initial important and deferred token identities. Both compressed policies remove 69.9409% of the multimodal tokens from the initial prefill by weighted count. RainCheck protects the first 16 generated tokens and then materializes at most one 31-token source-ordered chunk per decode step. `Hard` performs no deferred materialization and reuses the permanently pruned prefix at T2 and T3. RainCheck completes deferred materialization during T1 and reuses the complete prefix at T2 and T3. Table 4 shows T1; Table 25 reports T1, T2, and T3, with 288 observations per turn and model-policy pair. The complete workload contains 190 turns across 48 dialogues, including a fourth turn in 46 dialogues. Each model-policy pair therefore has 1,140 observations across all measured turns, including 852 continuation observations.

TTFT spans local request construction through the first generated token. For each request, mean TPOT divides the measured decode span by the 511 inter-token intervals. Reported turn-level means average the resulting request-level values. The request is the timing observation. Dialogue-cluster bootstrap resampling retains the three GPUs, two repetitions, all three or four turns, and paired policies for every sampled dialogue. Confidence intervals use 10,000 bootstrap draws with seed 3407. The per-request mean TPOT definition follows DistServe (Zhong et al., 2024).

Table 25 reports paired relative changes and confidence intervals by turn. `Hard` provides a matched pruning-only reference with a shorter context than `Ours` uses after T1.

Table 25: Paired SGLang timing comparisons. Entries report candidate-minus-baseline relative changes with 95% dialogue-cluster confidence intervals; negative values indicate lower latency.

Model	Turn	Metric	`Hard` vs. `Full`	`Ours` vs. `Full`	`Ours` vs. `Hard`
Qwen3-VL-8B	T1	TTFT	−21.64% [−24.97, −18.65]	−24.11% [−27.81, −20.15]	−3.16% [−7.09, +1.44]
	T1	TPOT	−0.91% [−1.14, −0.71]	+0.13% [−0.01, +0.28]	+1.05% [+0.72, +1.42]
	T2	TTFT	−0.09% [−0.84, +0.65]	−0.45% [−1.29, +0.36]	−0.36% [−1.04, +0.32]
	T2	TPOT	−0.75% [−0.89, −0.63]	+0.03% [−0.04, +0.10]	+0.79% [+0.66, +0.92]
	T3	TTFT	+0.46% [+0.07, +0.88]	+0.80% [+0.27, +1.43]	+0.34% [−0.29, +1.03]
	T3	TPOT	−0.50% [−0.72, −0.31]	−0.03% [−0.10, +0.05]	+0.47% [+0.28, +0.70]
Qwen3.5-9B	T1	TTFT	−16.87% [−19.86, −13.64]	−17.90% [−22.08, −13.08]	−1.23% [−4.98, +3.07]
	T1	TPOT	−0.48% [−0.56, −0.40]	+0.47% [+0.31, +0.65]	+0.95% [+0.74, +1.19]
	T2	TTFT	+0.02% [−0.48, +0.47]	−0.03% [−0.53, +0.47]	−0.05% [−0.50, +0.43]
	T2	TPOT	−0.26% [−0.37, −0.16]	+0.10% [+0.01, +0.19]	+0.36% [+0.26, +0.46]
	T3	TTFT	+0.27% [+0.02, +0.53]	+0.32% [−0.02, +0.77]	+0.06% [−0.32, +0.50]
	T3	TPOT	−0.19% [−0.28, −0.10]	+0.04% [−0.04, +0.12]	+0.23% [+0.14, +0.33]

The two `Ours`-versus-`Hard` T1 TTFT intervals contain zero, showing no clear first-turn TTFT difference in the paired experiment. Every `Ours`-versus-`Hard` TPOT interval is positive; `Hard`

omits deferred prefill and retains a shorter context. The T1 TPOT difference stays near 1% and decreases on the continuation turns.

Table 26: All-turn and continuation aggregates for the paired SGLang experiment. Each model-policy pair contains 1,140 all-turn observations and 852 continuations, including T4 when present. Policy columns report mean latency in milliseconds. The final columns report relative mean changes.

Model	Population	Metric	`Full`	`Hard`	`Ours`	`Ours` vs. `Full`	`Ours` vs. `Hard`
Qwen3-VL-8B	All turns	TTFT	56.245	46.965	45.919	−18.358%	−2.226%
	All turns	TPOT	6.278	6.231	6.279	+0.010%	+0.771%
	Continuations	TTFT	17.813	17.825	17.849	+0.201%	+0.134%
	Continuations	TPOT	6.308	6.263	6.306	−0.028%	+0.680%
Qwen3.5-9B	All turns	TTFT	69.788	62.523	62.090	−11.030%	−0.692%
	All turns	TPOT	6.526	6.508	6.536	+0.157%	+0.438%
	Continuations	TTFT	35.508	35.551	35.566	+0.163%	+0.041%
	Continuations	TPOT	6.537	6.523	6.540	+0.053%	+0.266%

Across all measured dialogue turns, `Ours` changes mean TPOT by 0.010% on Qwen3-VL and 0.157% on Qwen3.5 relative to `Full`. RAINCHECK completes deferred prefill in every T1 request and reuses the completed prefix without repeating deferred prefill on later turns. `Hard` performs no deferred prefill and reuses the permanently pruned prefix on every continuation.

Video first-turn measurements. Table 4 reports first-turn latency on MultiVerse and MT-Video-Bench; Table 25 includes later-turn MultiVerse comparisons from the same full-dialogue experiment. The MT-Video-Bench measurements use 48 fixed dialogues, eight per category, with 48 distinct available videos selected by a score-independent seed-3407 hash ordering. Each model runs six repetitions on one H20-3e GPU, with tensor parallelism one, one resident request, FlashAttention-3, and CUDA graphs. Each video supplies 64 native-processor frames with a maximum of 200,704 pixels per frame. All video policies use the same slow video processor, selection plan, and model-specific `Full` output trace. The first-turn traces average 196.96 tokens for Qwen3-VL and 146.88 for Qwen3.5. The systems configuration restores at most 31 deferred tokens per decode step after a 16-token guard; the Transformers quality experiments use 31-token chunks, as specified in Appendix C.2. The main video table uses first-turn measurements from the ready-prefix experiment; Table 28 reports a separate full-dialogue experiment that submits the next turn immediately after answer return. The two video experiments use the same cohort and output traces, but their timing observations are aggregated separately. Comparisons are paired within each dataset and model, with no pooling of image and video observations.

The frozen selection plans contain an average of 399.48 deferred tokens per image dialogue and 4,059.92 per video dialogue. The configured graph schedule requires, on average, 29.88 and 147.94 output positions to finish restoration, respectively, with maxima of 63 and 156 including the initial decode-only steps. The schedule-derived counts explain why the fixed 512-token image traces allow full restoration; completion for naturally terminated answers also depends on answer length. Video completion depends on both the larger deferred prefix and the shorter output traces.

End-to-end (E2E) response latency spans local request submission through the final output token, excluding subsequent restoration and external network or interface latency. For each request, E2E latency equals TTFT plus the number of inter-token intervals multiplied by request-average TPOT. The table averages the measured E2E latencies across requests. Post-response time spans the scheduler's response-completion boundary through complete-prefix readiness, not the client's readiness-polling duration. The mean and nearest-rank P95 include all requests, including requests that finish restoration during decoding. A sufficiently long application scheduling interval covers the remaining restoration; no fixed application interval is assumed.

Restoration interval measurement. MultiVerse active TPOT uses T1-only requests with token-arrival and restoration-event tracing, retaining the same 48 dialogues, 512-token traces, selection plans, and chunk/guard settings. Each model runs six repetitions on one GPU; the full-dialogue latency measurements use two repetitions on each of three GPUs without token-arrival tracing. T1-only requests omit session creation, and the two measurement populations are aggregated separately.

Active TPOT measures output intervals that contain foreground deferred prefill. We average those intervals within each request, then average the 288 request means. Guard intervals and restoration after the last output token are excluded; partial final chunks are included. With the 16-token guard, restoration event i, starting at zero, maps to inter-token interval $15 + i$, also starting at zero. The comparison policies use the same output positions and identical output traces. We check one output token per notification, per-step and cumulative restored-token counts, and complete request coverage. The `Full`/`Hard`/`Ours` means are 6.2116/6.1653/7.2010 ms on Qwen3-VL and 6.4584/6.4291/7.8817 ms on Qwen3.5. Raincheck finishes deferred prefill during the answer in all 288 requests per model. The small positive post-response intervals record readiness bookkeeping rather than unfinished visual-token processing.

E.2 Completion after Answer Return

Short-output completion stress test. The main 512-token SGLang workload completes deferred prefill during generation and therefore does not exercise post-response model execution. A separate 16-token Transformers experiment tests the completion mechanism when answer generation ends before restoration begins. The experiment uses the same 48-dialogue cohort with two repetitions and prevents restoration during generation through the 16-token guard. All 96 `Ours` T1 observations per model have deferred tokens awaiting prefill at answer readiness, and every later turn starts from the completed prefix. Table 27 reports the response boundary under blocking and cooperative completion. The 95% intervals use 5,000 paired dialogue-cluster bootstrap resamples.

Table 27: Independent 16-token Transformers completion stress test. Latencies are in milliseconds. The reduction compares cooperative with blocking response-ready latency.

Model	Blocking response		Cooperative response		Reduction	Response-to-prefix	
	Mean	P95	Mean	P95	Estimate [95% CI]	Median	P95
Qwen3-VL-8B	557.3	989.7	490.6	805.3	12.0% [10.3%, 13.6%]	43.6	222.5
Qwen3.5-9B	694.9	1213.4	626.5	1032.9	9.8% [8.8%, 10.9%]	81.5	216.9

The median and P95 pending deferred-token counts are 200 and 1,400 for both models. Cooperative completion moves the remaining prefill after response readiness and closes the prefix before the continuation request.

Immediate-follow-up video timing. We measure isolated, single-request execution on an H20 using SGLang, with one GPU per model. A score-independent cohort contains 48 dialogues from 48 videos, with eight dialogues per category and 284 turns. Each configuration runs six repetitions after three warmup dialogues, giving 288 first-turn and 1,416 continuation observations per model and configuration. Configuration order rotates within each dialogue and repetition. All configurations replay the same model-specific `Full` output tokens, including the recorded termination token, with exact token-sequence checks. For Qwen3.5, fixed-trace replay suppresses backend stop-token termination until the recorded sequence is complete. The timing backend uses 64 uniformly sampled frames with native resizing, a maximum of 200,704 pixels per frame, 30% initial visual retention, 31 deferred tokens per decode step, and a 16-token decode guard. The quality experiment uses the Transformers frame preprocessing and 31-token restoration steps in Appendix C.4.

`Ours` reduces first-turn TTFT relative to `Full` for both models, with first-turn TPOT increases from co-located restoration. Separating the 288 T2 observations from the 1,128 T3+ observations per configuration localizes the continuation delay to the first reuse of the video prefix. `Ours` increases mean T2 TTFT by 73.61 and 162.12 ms relative to `Full`, but mean T3+ TTFT is 18.00 versus 18.01 ms and 34.30 versus 34.35 ms for the two models. Pooling all continuation turns gives Raincheck means of 33.00 and 67.37 ms; the pooled values combine the T2 recovery wait with subsequent ready-prefix requests.

The T2 delay arises when deferred tokens remain unmaterialized at the end of the first answer. For Qwen3-VL-8B, 120 of 288 first-turn observations have residual tokens; for Qwen3.5-9B, 198 of 288 do. Across all first-turn observations, the mean residual counts are 566.4 and 1,430.6 tokens, respectively. The response-to-prefix interval measures the time from answer completion to readiness of the parent prefix for continuation. The mean intervals are 74.0 and 162.8 ms; the respective

Table 28: Video timing means on the fixed 48-dialogue cohort. First-turn TTFT is in seconds; all other entries are in milliseconds. T2 is submitted immediately after T1, with no imposed think time, and includes parent-prefix waiting. T3+ reports later turns separately. Each TPOT entry averages per-request mean inter-token intervals.

Model	Policy	T1 TTFT (s)	T1 TPOT (ms)	T2 TTFT (ms)	T3+ TTFT (ms)	Later TPOT (ms)
Qwen3-VL-8B	`Full`	3.569	6.680	18.15	18.01	6.759
	`Hard`	3.102	6.312	18.39	18.30	6.418
	`Ours`	3.107	8.014	91.76	18.00	6.762
Qwen3.5-9B	`Full`	3.480	6.631	34.80	34.35	6.647
	`Hard`	3.062	6.471	35.08	34.62	6.507
	`Ours`	3.063	7.622	196.92	34.30	6.653

medians are 0.1 and 197.4 ms, and the nearest-rank 95th percentiles are 357.5 and 371.8 ms over the 288 observations per model. The completed answer is released before the parent prefix becomes ready, so the residual recovery delays the dependent continuation instead of the completed answer.

E.3 SGLang Scheduling and Fusion Controls

We measure scheduling sensitivity on the same 48 MultiVerse dialogues with MMTok selection plans and prescribed 512-token first-turn traces. Each setting uses one H20-3e GPU, tensor parallelism of one, FlashAttention-3, CUDA graphs, page size one, and unchunked initial prefill. Both models use the slow image processor in the scheduling and fusion experiments. Each run keeps one engine resident, rotates the policy order, performs three dialogue warmups and warmups for all 48 input shapes, and measures two repetitions per dialogue and policy. Every request completes its prescribed trace, and Raincheck and the serial control materialize every deferred token. TPOT divides each request's decode duration by 511 token intervals. Percentage changes compare policy means within the same run; 95% intervals use 10,000 paired dialogue bootstrap resamples with seed 3407, retaining both repetitions for each sampled dialogue. The serving measurements use request-level timing without per-token CUDA events or additional synchronization. The SGLang timing experiments use separate measurement runs from the Transformers quality sweeps in Appendix D.5.

Chunk size. Table 29 varies c at 30.06% realized initial retention with a 16-token guard. For Qwen3.5-9B, increasing c from 7 to 31 reduces the TPOT increase relative to `Full` from 2.669% to 0.381%, with similar TTFT reductions. The Qwen3-VL-8B intervals at $c = 15$ and $c = 31$ overlap. The larger $c = 63$ setting increases mean TPOT on both models relative to $c = 31$, so larger chunks do not uniformly improve decode latency. From the deferred-token counts and graph capacities, the mean planned completion positions are 73.5, 43.6, 29.8, and 23.9 tokens for $c = 7, 15, 31, 63$; the respective maximum positions are 219, 112, 63, and 40. The planned positions follow the deterministic schedule, and Raincheck completes restoration in every measured request.

Table 29: First-turn chunk-size timing sensitivity. Changes are `Ours` relative to the same run's `Full` baseline; brackets give 95% paired intervals. Each policy has 96 observations from 48 dialogues.

Model	c	Mean TPOT (ms)			TTFT Δ (%)	TPOT Δ (%) [95% CI]
		`Full`	`Hard`	`Ours`		
Qwen3-VL-8B	7	6.237	6.184	6.261	−24.46	+0.386 [+0.187, +0.595]
	15	6.216	6.154	6.219	−25.00	+0.052 [−0.106, +0.224]
	31	6.212	6.154	6.222	−24.30	+0.157 [−0.001, +0.320]
	63	6.211	6.155	6.237	−24.09	+0.432 [+0.197, +0.689]
Qwen3.5-9B	7	6.491	6.465	6.664	−18.39	+2.669 [+1.985, +3.466]
	15	6.509	6.475	6.560	−18.50	+0.786 [+0.531, +1.071]
	31	6.499	6.469	6.524	−18.22	+0.381 [+0.206, +0.567]
	63	6.523	6.488	6.568	−18.26	+0.679 [+0.446, +0.941]

Initial retention. Table 30 fixes $c = 31$ and varies realized initial retention from 30.06% to 70.08%. Lower retention yields larger TTFT reductions, because fewer visual tokens enter initial prefill. At 70.08% retention, mean TTFT decreases by 10.77% on Qwen3-VL-8B and 8.83% on Qwen3.5-9B relative to `Full`. The Qwen3-VL-8B TPOT interval at 70.08% includes zero. The 30.06% rows reuse the $c = 31$ runs in Table 29.

Table 30: First-turn retention timing sensitivity at $c = 31$ and a 16-token guard. Retention is the fraction of visual tokens retained, pooled over the cohort. Intervals follow Table 29.

Model	Retention (%)	Mean TPOT (ms)			TTFT Δ (%)	TPOT Δ (%) [95% CI]
		`Full`	`Hard`	`Ours`		
Qwen3-VL-8B	30.06	6.212	6.154	6.222	−24.30	+0.157 [−0.001, +0.320]
	50.01	6.228	6.183	6.231	−17.32	+0.052 [−0.091, +0.197]
	70.08	6.228	6.188	6.222	−10.77	−0.103 [−0.237, +0.032]
Qwen3.5-9B	30.06	6.499	6.469	6.524	−18.22	+0.381 [+0.206, +0.567]
	50.01	6.496	6.464	6.507	−13.69	+0.160 [+0.040, +0.288]
	70.08	6.493	6.470	6.500	−8.83	+0.117 [+0.032, +0.207]

Fused versus serial materialization. The four-policy control fixes $c = 31$, initial retention at 30.06%, and the guard at 16 tokens. The serial control materializes deferred tokens and then decodes the next token on the same CUDA stream, using the same restoration plan as fused Raincheck. Table 31 compares complete restoration under the two execution schedules. Fusion reduces mean TPOT relative to serial execution by 5.916% on Qwen3-VL-8B (95% CI [4.657%, 7.309%]) and 12.932% on Qwen3.5-9B ([10.307%, 15.781%]). The comparison measures fusion together with the associated scheduling change. Relative to `Full`, the TPOT changes for fused `Ours` are +0.168% [+0.036%, +0.313%] and +0.349% [+0.201%, +0.507%], respectively. The fused-versus-serial TTFT intervals include zero for both models.

Table 31: Four-policy first-turn fusion control, measured in a separate batch with 48 dialogues and two repetitions per policy. Both Raincheck schedules restore all deferred tokens.

Model	Policy	Mean TTFT (ms)	Mean TPOT (ms)
Qwen3-VL-8B	`Full`	163.135	6.228
	`Hard`	125.963	6.176
	`Ours`, fused	125.648	6.238
	`Ours`, serial	121.181	6.630
Qwen3.5-9B	`Full`	164.611	6.479
	`Hard`	136.965	6.452
	`Ours`, fused	137.321	6.501
	`Ours`, serial	132.222	7.467

F Motivation Experiments

F.1 Query-Dependent Token Selection

Figure 4(a) compares independently computed MMTok selections for the first two questions of 48 MultiVerse dialogues. For each dialogue, the missing fraction is $|K_2 \setminus K_1|/|K_2|$, where K_1 and K_2 are the selected multimodal-token sets. Each curve reports the fraction of dialogues whose missing fraction exceeds the horizontal-axis threshold. Figure 3 uses two illustrative questions on a TextVQA image and an independently annotated jersey-number region. The measured Qwen3-VL-8B mask retains 36 of 117 tokens on a 13×9 grid and illustrates the input positions omitted by pruning.

Answer-conditioned selection. Adding the completed answer changes which tokens MMTok selects. We compare the first question alone with the question followed by the same model's completed `Full` answer on the same 48 MultiVerse dialogues. Both conditions use identical image features, FP32 normalization, and per-image 30% retention counts. Let K_Q and K_{QA} denote the two

selected-token sets. We measure $100|K_{QA} \setminus K_Q|/|K_{QA}|$, the percentage of question-and-answer selections absent from the question-only selection. Figure 4(b) shows the distribution across dialogues. Table 32 reports equal-dialogue means and 95% percentile confidence intervals from 10,000 dialogue bootstrap resamples with seed 3407.

Table 32: Answer-conditioned selection differences at 30% retention. Values report the percentage of answer-conditioned selections absent from the question-only selection. High-quality answers satisfy every reference checklist item and receive a quality score of at least 7/10.

	All answers		High-quality answers	
Model	n	Mean [95% CI]	n	Mean [95% CI]
Qwen3-VL-8B	48	47.14 [43.55, 50.74]	12	41.95 [35.59, 48.28]
Qwen3.5-9B	48	42.48 [38.35, 46.61]	11	39.39 [32.26, 47.61]

The retrospective comparison uses unchanged, previously generated `Full` answers in the format `Question: ... Answer: ...`. Adding the answer changes both text content and length. Answer length correlates positively with the selection difference, with Pearson coefficients of 0.190 and 0.244 for Qwen3-VL and Qwen3.5, respectively. Answers reaching the 512-token generation cap show larger differences than uncapped answers. The high-quality subsets retain substantial selection differences but do not isolate the effect of answer length. The diagnostic measures how the available text changes token selection, without testing whether the newly selected tokens are necessary to answer the question. Answer-quality gains from deferred prefill are evaluated separately in Section 3.2 and Appendix D.1.

F.2 Deferred-Token Budget

Figure 5 uses a separate single-request SGLang experiment on one H20-3e GPU with tensor parallelism of one, FlashAttention-3, and CUDA graphs. We select eight dialogues from the 48-dialogue cohort by evenly spaced ranks in deferred-token count among dialogues with at least 255 deferred tokens in both models. Each configuration has four repetitions per dialogue and shares the MMTok partition, 30% initial retention, 16-token guard, and fixed 512-token output trace across policies. Both models use the slow image processor with the same pixel limits as the systems profile.

We vary the deferred-token budget over $\{1, 3, 7, 15, 31, 63, 127, 255\}$ and timestamp output arrivals at the tokenizer manager. For each request, we average only intervals that process a full-size deferred chunk and compare decode-only intervals at the same output positions. The plotted value averages the 32 request-level means; smaller terminal chunks and post-response completion are excluded. The horizontal axis is logarithmic in the total tokens per step, equal to the deferred budget plus one decode token. The measured intervals include scheduling and output delivery as well as model execution.

Smaller budgets cover more output positions, so the curves describe natural restoration trajectories with growing context rather than fixed-context kernel timings. Any deferred tokens left after the fixed output trace are processed by completion prefill. The restoration-interval measurements in Figure 5 use a different population from the whole-request mean TPOT in Table 4.

F.3 Inter-Turn Completion Windows

Multimodal message intervals. The interval is the next user message creation time minus the preceding assistant message creation time. Because the assistant timestamp precedes final-token receipt, the message interval can include answer-generation time. We sample 2,000 ShareChat conversations containing a user, assistant, and subsequent user message using seed 3407 (Yan et al., 2026). The original share pages are available for 1,855 conversations; matching their multimodal metadata to the released timestamps identifies 118 multimodal conversations and 119 positive intervals. Excluding conversations with any matched interval below one second, including negative intervals, leaves 102 observations. Table 33(a) reports interval quantiles; panel (b) gives the complete duration bins. Figure 6(a) combines the three bins of at least five minutes for readability.

Tool-call latency. TraceLab reports coding-agent traces from Claude Code and Codex (SyFI TraceLab, 2026). The timing analysis uses strictly positive individual tool latencies, preferring internal duration and otherwise using the elapsed time from tool emission to result. The published distribution has a median of 0.3 s, a 90th percentile of 13.5 s, and a 99th percentile of about 3 min. Table 33(c) shows the published duration bins used in Figure 6(b). The tool-call percentages sum to 100.1% because the published values are rounded. Individual tool-call durations can overlap and therefore do not equal elapsed time between consecutive model requests. We report tool-call latency separately from the multimodal message intervals.

Table 33: ShareChat multimodal follow-up message intervals and TraceLab tool-call durations. (a) ShareChat quantiles. (b) Duration bins for the 102 cleaned ShareChat observations. (c) Published TraceLab positive tool-call durations.

(a) ShareChat multimodal follow-up intervals

Cleaning unit	Count	P5 (s)	Median (s)	P75 (s)	$\geq$60 s
Conversation	102	18.1	101.4	387.6	72.5%

(b) ShareChat duration bins

Interval	1 to 10 s	10 to 60 s	1 to 5 min	5 to 30 min	30 min to 2 h	$\geq$2 h
Count	3	25	44	15	8	7
Share	2.9%	24.5%	43.1%	14.7%	7.8%	6.9%

(c) TraceLab individual tool-call durations

Duration (s)	<1	1 to 10	10 to 60	>60
Share of calls	61.3%	25.4%	10.0%	3.4%

Interpretation. Whether completion finishes within an application interval depends on the deferred remainder and available scheduler capacity. An early follow-up waits at the completion barrier before context reuse.

G Evaluation Scope and Deployment Extensions

G.1 Evaluation Scope

The multi-turn quality results use a 512-token per-turn budget and dataset-specific judges, as specified in Appendix C. Appendix D.3 defines score aggregation and statistical uncertainty.

RAINCHECK optimizes LLM prefill and leaves modality encoding unchanged. Evaluation covers Qwen3 full attention and Qwen3.5 GDN. The systems backend uses one running request with tensor parallelism of one, as specified in Appendix E.1. The measurements establish the co-located execution tradeoff at batch size one; other recurrent architectures, batched serving, and speculative decoding require separate evaluation. The recorded-factor approximation is analyzed in Appendix B.5.

Batch size and execution capacity. Deferred prefill changes when input tokens are processed, not the need to process the recovered tokens. As decode batches consume more compute capacity, deferred tokens can increase iteration time or require a smaller restoration budget. A smaller budget slows recovery and can leave more input tokens to process after answer return. The state-update rules apply per request, but batched execution additionally requires request-specific checkpoints, attention views, and recurrent maps. The main systems configuration allows at most 31 deferred tokens alongside one decode token.

Speculative verification. SpecVLM reduces draft-model computation through verifier-guided video token pruning (Ji et al., 2025), while LVSpec uses visual-semantic guidance to relax verification (Ji et al., 2026). A possible integration would hold the materialized context fixed during each draft-and-verification round and commit deferred updates between rounds. Only accepted outputs would enter the retained KV cache and recorded recurrent transitions; rejected candidates would require rollback of tentative states. Verification and deferred prefill could share dense projections,

but require different attention views and recurrent-state handling. Draft generation, acceptance rate, state rollback, and the context read by verification determine latency in addition to the number of token rows. Compatibility requires preserving the chosen verification rule for the fixed context in each round; the resulting process targets Raincheck's evolving context, not equivalence to full-prefill generation.

G.2 PD-Disaggregated State Join

In a PD-disaggregated design, a separate prefill worker completes deferred prefill from the input checkpoint. The decode worker retains the generated-token KV cache, $P_Y \in \mathbb{R}^{d_k \times d_k}$, and the compact prefix and live states in $\mathbb{R}^{d_k \times d_v}$. The prefill worker starts from the input checkpoint, materializes D in source order, and returns the completed input state and deferred KV entries. With the recorded generated-token factors held fixed, the decode worker forms

$$\begin{aligned} \mathrm{KV}_{\mathrm{reuse}} &= [\mathrm{KV}_{I_0} \mid \mathrm{KV}_D \mid \mathrm{KV}_Y], \\ S_{\mathrm{reuse}} &= S_{\mathrm{live}}^{\mathrm{compact}} + P_Y \left(S_{\mathrm{prefix}}^{\mathrm{complete}} - S_{\mathrm{prefix}}^{\mathrm{compact}} \right). \end{aligned} \tag{16}$$

All recurrent states in Equation 16 have shape $d_k \times d_v$. The attention path inserts deferred entries before the retained generated region. When insertion changes the short-convolution history, Equation 14 replaces the prefix-state difference and P_Y with the repaired-anchor difference and the remaining suffix matrix.

Turn-boundary recovery. One placement keeps the compact context fixed throughout the first answer and joins recovered input states after answer completion. Later turns can use the recovered context, but the first answer receives no quality benefit from deferred-token recovery. Joining states during generation could expose recovered input earlier, at the expense of additional synchronization. The prefill worker processes deferred tokens separately from decode; the state join preserves the attention dependencies and recorded-factor mapping. Separate workers require additional prefill capacity and state transfer (Zhong et al., 2024). Boundary repair and the final join still execute on the decode side. Prefix reuse waits for worker completion, state transfer, and the final join, so an early follow-up can incur a recovery delay.

H Extended Related Work

H.1 Multimodal Token Pruning

Pruning can follow the vision encoder (Zhang et al., 2024a; Liu et al., 2025) or early LLM layers (Chen et al., 2024). Layer-wise methods adapt the retained set across LLM layers (Xing et al., 2025; Ye et al., 2025). Selection criteria include diversity in DivPrune, multimodal coverage in MMTok, and representation changes in TransPrune (Alvar et al., 2025; Dong et al., 2026; Li et al., 2026). Query relevance can guide pruning inside the vision encoder, as in QuietPrune, or inside the LLM, as in SparseVLM (Gao et al., 2026; Zhang et al., 2025b). OmniDelta allocates a fixed token budget across audio and video according to query intent, then distributes each modality's budget over segments or frames using input content (Huang et al., 2026).

H.2 Information Retention and Recovery

SparseVLM and VisionZip aggregate information from pruned tokens (Zhang et al., 2025b; Yang et al., 2025a). MetaCompress learns an image-conditioned, prompt-agnostic compressed representation for reuse across turns (Wang et al., 2026). SparseVILA retrieves query-relevant states from a retained visual cache (Khaki et al., 2025). DyCoke retains pruned visual KV entries in a separate cache and brings entries back into attention as relevance changes during generation (Tao et al., 2025). DSTP keeps excluded visual tokens in reserve and temporarily adds newly relevant tokens during generation (Kim et al., 2026). LazyLLM caches intermediate hidden states and computes missing KV states on demand when pruned tokens are selected again (Fu et al., 2024). Raincheck schedules deferred multimodal tokens for prefill in source order, progressively expanding the context available to the ongoing answer. The evaluated policy completes the input prefix before reuse by later turns.

H.3 Execution and Hybrid Architectures

HMI combines hierarchical knowledge management with parameter swapping and prefetching for resource-efficient multi-tenant inference (Zhang et al., 2025a). Within the serving engine, PagedAttention supports flexible KV-cache allocation for batched requests (Kwon et al., 2023). Prefill and decode can also share execution: Sarathi and Sarathi-Serve batch prefill chunks with decoding requests (Agrawal et al., 2023; 2024). Sarathi fuses prefill and decode linear computations to share weight accesses, and computes attention separately. POD-Attention overlaps prefill and decode within the attention operator, and Libra partitions requests into schedulable micro-requests (Kamath et al., 2025; Ruan et al., 2026). RAINCHECK co-schedules prefill and decode within the same request and maintains the state dependencies introduced by deferred input tokens.

Hybrid models such as Qwen3.5 combine full attention with Gated DeltaNet (GDN) layers (Qwen Team, 2026). GDN uses a scalar decay gate with a delta-rule update, and Kimi Delta Attention (KDA) introduces finer-grained channel-wise gating (Yang et al., 2025b; Kimi Team, 2025). Both architectures support chunkwise evaluation of recurrent state updates. Full attention caches per-token keys and values; recurrent layers summarize the prefix in a fixed-size state. Deferred input therefore requires updating the recurrent state as well as attention visibility.