MentorPulse: Refreshing Cross-Model Latent Guidance for Long-Form Generation

Ziwu Liu Guozhong Li Chen Qiu Weiyang Kong Panos Kalnis
King Abdullah University of Science and Technology (KAUST)

Abstract

Cross-model latent guidance lets a frozen large mentor encode an input once and a frozen small student generate from the resulting signal. Existing methods keep this signal fixed, assuming it stays useful as the output grows; we show this fails in long-form generation. On multi-turn instruction following, static guidance pushes a 4B student's constraint satisfaction 2.5 points below its no-guidance baseline; a training-free refresh every 16 tokens changes only the memory content and restores a 2.0-point gain over that baseline. We propose MentorPulse to keep guidance fresh at practical cost: it compresses mentor states into a capped slot memory, incrementally processes newly generated tokens, and updates the memory that the student reads through gated cross-attention without resetting the student's KV cache. Windowed Refresh Training exposes the bridge to prefix-conditioned memory. Across thirteen datasets, MentorPulse closes 52.2% of the mentor–student gap on macro average, outperforming C2C, T2T, and equal-budget LoRA, with the largest gains on long outputs. It performs best on all eleven mentor–student pairs from three model families, with margins that narrow as the capability gap grows, and a lightweight read-pattern check predicts the gain before deployment. Measured costs identify refresh intervals that dominate text guidance on long outputs.

1 Introduction

A frozen large *mentor* can guide a frozen small *student* through latent states [Bergner et al., 2024, Fu et al., 2026, Chen et al., 2026]. Existing methods compute this guidance before decoding and keep it fixed. In long-form generation this design fails: as the generated prefix grows, prompt-only guidance can become stale and push the student below its no-guidance baseline, while fresh, prefix-conditioned guidance reverses the degradation without changing the student (Figure 1). We ask how to keep latent guidance fresh throughout decoding at practical cost.

Model capability and serving cost both grow with scale [Kaplan et al., 2020, Wang et al., 2025, Chen et al., 2025]. Letting a mentor process the input while a student generates

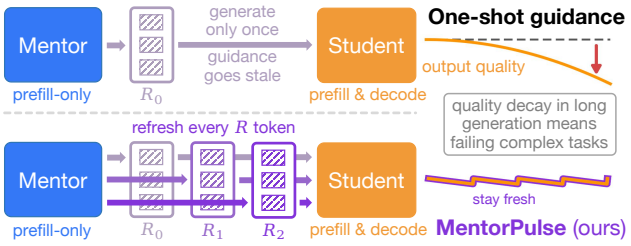

Figure 1: One-shot guidance fixes the memory before decoding and quality decays over long generation (top); MentorPulse refreshes it from the generated prefix every R tokens (bottom).

retains useful large-model information at reduced decoding cost; long-form generation offers the largest saving because it has the most decoding steps, yet there fixed guidance is most likely to become outdated. A compact, refreshable memory further extends prefill–decode disaggregation [Zhong et al., 2024b, Patel et al., 2024] across model sizes and networked devices.

Prior methods establish that a student can use mentor states through a learned interface: LLM-to-SLM transfers a one-time prompt encoding [Bergner et al., 2024], C2C fuses the mentor's KV cache into the student's cache [Fu et al., 2026], and Latent-Guided Reasoning transfers compact strategy vectors [Chen et al., 2026]. All keep the signal fixed during student decoding—usually harmless for short outputs, but untested systematically for long outputs, where the generated prefix changes the relevant context.

Section 3 isolates freshness with a training-free intervention: at fixed intervals the mentor prefills the input and generated prefix, then replaces the memory, leaving the student's weights and KV cache unchanged. On affected long-output tasks, static guidance falls below the student-only baseline and fresh guidance restores a gain; fresh but mismatched memory does not help, attributing the repair to updated content. Tasks beyond the student's capability do not improve, and short outputs finish before the first update. The diagnostic thus separates staleness from capability limits, yet reprocessing the growing prefix at every update makes it too expensive to deploy.

MentorPulse makes this repair efficient. It compresses mentor states into a capped, position-aware slot memory that the frozen student reads through gated cross-attention. Versioned incremental refresh reuses the mentor's KV cache to process only new tokens, then replaces the memory without invalidating the student's decoding history; Windowed Refresh Training optimizes the output window of each sampled memory version. The interval R controls the quality–cost trade-off, and $R \to \infty$ recovers one-shot guidance.

Across thirteen datasets, MentorPulse closes 52.2% of the mentor–student gap on the main pair, about twice the recovery of the strongest prior alternative. Its advantage is concentrated on long outputs, where static guidance becomes harmful, and it performs best on all eleven pairs from three model families; the margin narrows on the most capability-separated pairs, tracked in advance by our read-distribution indicator V_{64}. Measured throughput places the default $R{=}16$ among the cost-efficient settings $\{8, 16, 32\}$ against T2T on long-output tasks.

Our contributions are: (1) we identify guidance staleness as a failure mode of one-shot guidance and separate it from student capability limits via a training-free diagnostic; (2) we introduce MentorPulse, combining versioned slot memory, incremental mentor prefill, and Windowed Refresh Training without changing the student's decoding cache; (3) we demonstrate generality across eleven pairs and three model families, and propose the 64-step read-distribution variance V_{64} to screen pairs before deployment; (4) we provide byte-level refresh accounting and measure when MentorPulse is more cost-efficient than text guidance on long-output tasks. MentorPulse thus turns long-form generation from the regime where one-shot guidance fails into the one where latent guidance pays off most.

2 Related Work

One-shot cross-model guidance. The closest predecessors transfer input-conditioned latent state across models. LLM-to-SLM encodes the prompt once with a frozen large model and conditions a smaller model on the projected representations [Bergner et al., 2024]. C2C projects and fuses a Sharer's input KV cache with a Receiver's cache layer by layer [Fu et al., 2026], and Latent-Guided Reasoning compresses a large model's solution strategy into a fixed set of latent vectors for a smaller reasoner [Chen et al., 2026]; related single-model work compresses prompts or context into latent summaries [Lester et al., 2021, Mu et al., 2023, Ge et al., 2024, Chevalier et al., 2023]. The transferred objects differ, but the schedule is shared: the signal is set before the receiving model decodes and is never revised from its evolving output. MentorPulse takes the step LLM-to-SLM names as a future direction, refreshing the transferred signal from the generated prefix while preserving the student's autoregressive state; among these interfaces, only MentorPulse combines a frozen mentor, an explicit state-count cap, and prefix-conditioned refresh (Table 6, appendix).

Other large–small model collaboration. Distillation transfers teacher behavior into student parameters, and LoRA adapts a restricted weight subset; both yield persistent parameters rather than input-specific latent state [Hinton et al., 2015, Jiao et al., 2020, Hu et al., 2022, Magister et al., 2023]. Routing and cascading conditionally invoke one or several models but exchange selections or text, not hidden state consumed by a continuously decoding student [Zhang et al., 2025, Jitkrittum et al., 2025, Chen et al., 2024, Wang et al., 2025, Chen et al., 2025]. Speculative decoding is lossless acceleration: the target model verifies cheap drafts and preserves its own distribution, at the price of engaging both models at every step [Leviathan et al., 2023, Chen et al., 2023, Li et al., 2024, Hu et al., 2025]. MentorPulse instead leaves generation to the student while the mentor supplies intermittent prefix-conditioned state.

Dynamic state within one model. Prior work also updates state during inference, but produces and consumes it within one model: RefreshKV rebuilds a partial KV cache with occasional full-attention steps during long generation [Xu et al., 2025], and MemoryLLM updates a fixed-size latent memory from new text [Wang et al., 2024a]. These address attention efficiency and knowledge updating; MentorPulse refreshes guidance produced by a separate mentor while leaving the student's own KV cache untouched.

Disaggregated serving. Prefill–decode systems place the two inference phases of one model on different workers and ship request-specific KV state between them [Zhong et al., 2024b, Patel et al., 2024, Qin et al., 2025]; the unit of disaggregation is a phase of one model, not semantic guidance between different models. MentorPulse instead uses a boundary across model scales: a mentor produces semantic guidance that a separate student consumes, over a capped, prefix-independent payload. All these lines leave open the behavioral question central to this paper—can fixed input-only guidance become stale enough to harm the student as its generated prefix grows?—which Section 3 tests directly.

3 Guidance Staleness in Long-Form Generation

3.1 Setup

To test whether input-only guidance stays useful throughout long-form decoding, we pair a frozen Qwen3.5-27B mentor with a frozen Qwen3.5-4B student (A2) [Qwen Team, 2026]. The mentor prefills the input x once, its states are compressed into a slot memory Z, and the student reads Z at every layer through a gated cross-attention bridge. Only the bridge is trained; with all gates at zero the system reduces to

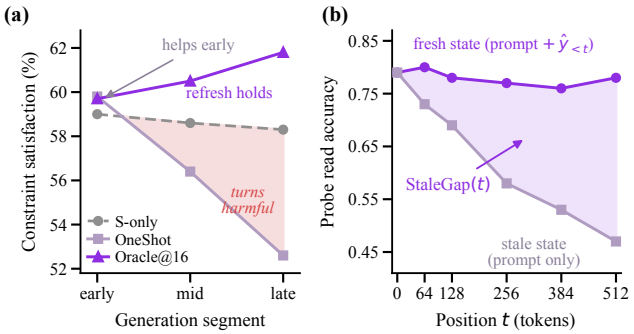

Figure 2: Staleness diagnosis on Multi-IF. (a) Effect: constraint satisfaction (fraction of verifiable constraints satisfied) by generation segment. (b) Mechanism: probe read accuracy for upcoming-window properties; the shaded wedge is $\mathrm{StaleGap}(t)$.

the plain student. We compare OneShot, the static bridge, against S-only, the student alone as a floor, and M-only, the mentor alone as a ceiling. Five diagnostic tasks, disjoint from the Section 5 benchmarks and bridge training data, are chosen for expected staleness sensitivity: Multi-IF [He et al., 2024b], QMSum [Zhong et al., 2021], and HelloBench long-form writing [Que et al., 2024] as positives, BigCodeBench [Zhuo et al., 2025] and ARC-Challenge [Clark et al., 2018] as controls. Prior work established that mentor states carry information the student cannot form alone [Bergner et al., 2024, Fu et al., 2026, Chen et al., 2026]; we ask how long it stays valid.

3.2 Staleness and the Oracle

One-shot guidance is computed from x at $t{=}0$, while the decision at step t is conditioned on $x \oplus \hat{y}_{<t}$. To quantify this mismatch, let q_t denote verifiable properties of a fixed-length window starting at position t, and let $A_{\mathrm{stale}}(t)$ and $A_{\mathrm{fresh}}(t)$ be the accuracies of matched linear probes reading q_t from $H_{\mathcal{M}}(x)$ and $H_{\mathcal{M}}(x \oplus \hat{y}_{<t})$. We define $\mathrm{StaleGap}(t) = A_{\mathrm{fresh}}(t) - A_{\mathrm{stale}}(t)$. The probes act on mentor states directly, without the bridge (Appendix G.4). On Multi-IF, A_{stale} falls from 0.79 at $t{=}0$ to 0.47 at $t{=}512$, while A_{fresh} stays between 0.76 and 0.80 (Figure 2b): the guidance drifts away from the task it describes.

Representation drift does not itself prove behavioral harm, so we intervene with a condition changing only freshness. OracleSwap@16 pauses decoding every 16 generated tokens, re-prefills $x \oplus \hat{y}_{<t}$ with the mentor, and swaps the rebuilt memory in place; the student's weights, generated text, and KV cache are untouched, and nothing is trained. *Oracle* means full recomputation, not access to labels or future tokens. On Multi-IF, OneShot scores 56.2, below the S-only floor of 58.7 by more than twice the pooled standard deviation: static guidance actively misleads a student trained to trust the memory. OracleSwap@16 lifts the score to 60.7, 4.5 above OneShot and 2.0 above the floor; QMSum and HelloBench move the same way (+2.7 and +3.1 over OneShot). OneShot helps early, crosses below the floor mid-generation, and collapses late, while OracleSwap@16 holds the late segment (Figure 2a). Because the swap changes nothing but the memory content, the repair can only come from freshness; equally fresh memory from a different sample brings no repair (Appendix B.1). The effect has boundaries: on BigCodeBench all conditions sit within noise, a capability failure that refresh cannot fix, and on ARC-Challenge outputs end before the first swap, so OracleSwap@16 equals OneShot exactly; Section 5.4 maps these boundaries across pairs.

The oracle repairs the failure but cannot be deployed: every swap re-prefills the entire growing prefix, so the cost of staying fresh grows with generation. A practical mechanism must process only new tokens, deliver updates without invalidating the student's decoding history, and train the bridge for mid-generation memory changes; Section 4 meets all three.

4 MentorPulse

The oracle of Section 3.2 shows what freshness is worth; MentorPulse delivers it under deployment constraints. Five commitments shape the design: both backbones stay frozen and only a pair-specific bridge is trained; the mentor prefills and never decodes; guidance lives in a slot memory separate from the student's KV cache, so an update replaces one tensor; the slot count is capped, giving every transfer a closed-form byte count; and a single interval R sets how often guidance is renewed. Figure 3 shows the flow.

4.1 Slot Memory Constructor

At $t{=}0$ both models prefill the input x, and the Slot Memory Constructor (SMC) turns the mentor's residual-stream states into slots (①–③). The newest $W_{\mathrm{tail}}{=}32$ positions are kept as raw states, preserving exact wording near the context boundary. Earlier positions are split into $S{=}32$-token segments, each contributing the true state at its final position rather than a pooled average (Table 1, row 9); at most $P{=}128$ prompt segments are kept, longer inputs getting evenly coarser segments. For each mentor layer o, states are normalized by a fixed scalar rms_o and projected from the mentor width d_m to the memory width d_t (equal to the student hidden size d_s) by a shared map with a rank-64 per-layer correction; embeddings tag mentor layer, slot order, and segment type, and a learned weight ρ_o scales each layer's slots; sorting layers by ρ_o sets the transmission pruning order (row 11). With k transmitted layers and $p_x \le P$ prompt segments, the initial memory is $Z_0 \in \mathbb{R}^{k(p_x+W_{\mathrm{tail}})\times d_t}$. Two properties matter downstream: the capped slot count keeps a transfer from growing with

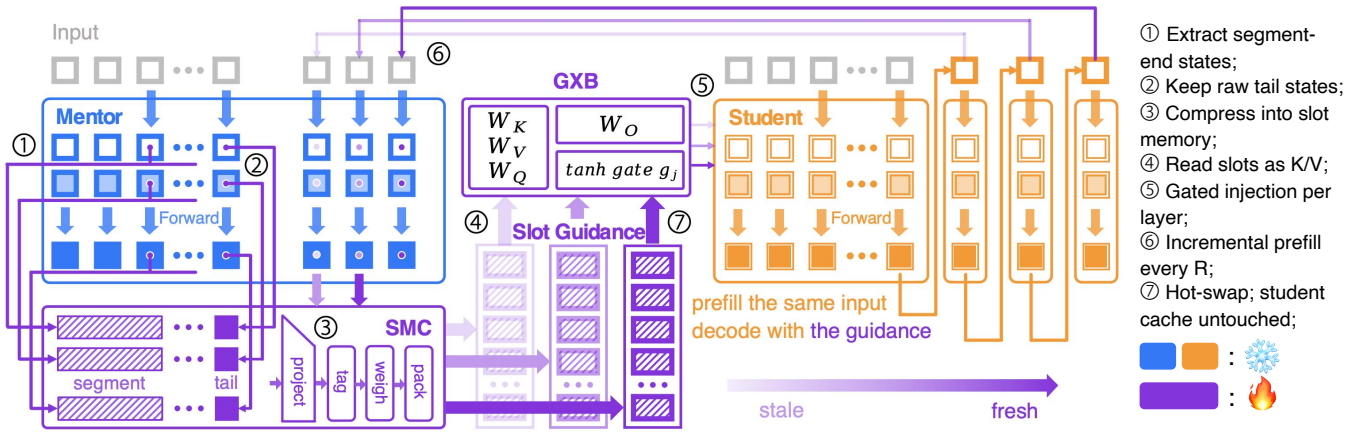

Figure 3: MentorPulse data flow. The frozen mentor (blue) prefills, the SMC compresses its states into slot guidance, and the frozen student (orange) reads the slots through the gated bridge; every R tokens an incremental prefill refreshes the memory in place (steps ①–⑦ in the text). Purple components are trained; saturation encodes freshness.

input length as a full KV cache would, and position-based segmentation needs no cross-model token alignment.

4.2 Gated Cross-Attention Bridge

The student reads the current memory after each layer j through gated cross-attention (④–⑤), $h_j \leftarrow h_j + \tanh(g_j)\,\mathrm{CrossAttn}(Q{=}h_j,\ K,V{=}Z_t)$: the query is the student residual stream, the memory supplies keys and values, and all four attention projections are low rank ($r{=}256$). The gates start at zero, so the untrained bridge is an identity map and the system reproduces the plain student bit for bit—the degeneracy check of Sections 3 and 5.3. Training escapes this dead zone by warm-starting from a static-bridge checkpoint whose gates are already open (Appendix E). Because Z_t is a side tensor outside the student's self-attention cache, replacing it never invalidates cached states—the basis of refresh.

4.3 Versioned Incremental Refresh

Every R generated tokens (⑥), the student sends its newest tokens to the mentor, which extends its own KV cache and computes states only for the new positions; by causality, earlier states are unchanged. Across a whole generation each token is therefore prefilled exactly once, and the mentor's total compute is independent of R, a fact the cost analysis of Section 5.6 builds on. One caveat: new queries still attend to the cached prefix, so a single refresh's attention cost can grow with accumulated context length.

The memory updates incrementally: the span covered by the previous tail is re-segmented into generated-prefix summary slots, a fresh tail is appended, and existing summary slots are never resent. The segment-type embedding distinguishes prompt summaries, generated-prefix summaries, and the current tail (row 12). The amortized bf16 payload per update is

$$\mathrm{Bytes}(R) = k\,(R/S + W_{\mathrm{tail}})\,d_t\,b, \tag{1}$$

with b bytes per value: a new summary slot completes only every $\lceil S/R \rceil$-th refresh, so R/S is the average summary payload per update, and the worst single update ships $\lceil R/S \rceil$ summary slots. For fixed R, k, S, and precision, the payload is independent of the accumulated prefix length and dominated by the tail slots. For A2 at $R{=}16$ the update is 10.6 MB with all 64 mentor layers ($k{=}64$) and 2.7 MB with the top 16 ($k{=}16$, row 11). At the j-th refresh the student swaps $Z_{t_{j-1}}$ for Z_{t_j} between two decoding steps (⑦); its weights, generated text, and self-attention cache stay untouched. $R \to \infty$ recovers the static bridge.

4.4 Windowed Refresh Training

A bridge trained only on fixed memory has never seen its guidance change; Section 5.3 (row 3) shows what that costs. Windowed Refresh Training (WRT) closes the gap while preserving parallel teacher forcing. For a pair (x, y), one mentor pass over $x \oplus y$ contains, by causality, the states of every refresh version. Each training row samples a boundary $t \in \{0, R, 2R, \ldots\}$, builds Z_t from the prompt and the label prefix $y_{\le t}$, and applies cross-entropy, over the bridge parameters θ only, to the window $(t, t{+}R]$, the span that decodes under Z_t at deployment:

$$\mathcal{L}_{\mathrm{WRT}} = -\mathbb{E}_{(x,y),t} \sum_{i=t+1}^{\min(t+R,|y|)} \log p_\theta(y_i \mid x, y_{<i}, Z_t). \tag{2}$$

Two approximations remain: prefix positions read the single latest Z_t although deployment decoded them under older versions, and label prefixes are a train–test mismatch

because inference conditions on student-generated prefixes (Appendix D). Sampling $t{=}0$ keeps static training in the objective, and WRT warm-starts from the static checkpoint with new embeddings zero-initialized. MentorPulse needs no mentor decoding at inference; labels are generated offline.

5 Experimental Evaluation

We ask whether refreshed latent guidance outperforms the student-only, static-latent, text-guidance, and parameter-adaptation baselines across input and output lengths; which components produce the gains and whether they generalize across mentor–student pairs; and how the refresh interval trades off quality, communication, and serving cost.

5.1 Experimental Setup

Hardware settings. The serving topology matches the deployment that Section 5.6 prices: the main-pair mentor runs on one data-center GPU (NVIDIA H100 80GB) and the student on one consumer GPU (NVIDIA RTX 4090 24GB), connected cross-site at an effective 150 Mbps with 18 ms round-trip latency. Both models are served with vLLM 0.19.1 in bf16; every throughput entering the cost model is measured on this stack at batch size 1 with an 8K-token context (measured rates, rental prices with quotation dates, and network and billing sensitivity analyses in Appendix H). All compared methods use the same serving stack and decoding configuration within each pair; the one registered exception is the Gemma-4 pair, served through HuggingFace transformers rather than vLLM for every condition alike (Appendix A.3).

Models and benchmarks. We evaluate eleven mentor–student pairs from three model families (Qwen, Gemma-4, and Ministral 3), with 1.7B–12B students; pairs sharing a mentor form a *mentor group* (A–E), and the main pair A2 is Qwen3.5-27B $\rightarrow$ Qwen3.5-4B. Thirteen datasets cover short-input selection (A–C: MMLU-Pro [Wang et al., 2024b], GPQA [Rein et al., 2024], AGIEval-MCQ [Zhong et al., 2024a]), short-input generation (D–H: MATH-500 [Hendrycks et al., 2021, Lightman et al., 2024], OlympiadBench [He et al., 2024a], LiveCodeBench [Jain et al., 2025], IFEval [Zhou et al., 2023], WritingBench [Wu et al., 2025]), long-input selection (I–J: LongBench v2 [Bai et al., 2025a], QuALITY [Pang et al., 2022]), and long-input long-output generation (K–M: GovReport [Huang et al., 2021], MultiNews [Fabbri et al., 2019], LongBench-Write [Bai et al., 2025b]).

Compared methods. S-ONLY and M-ONLY, the student and mentor alone, define the two anchors. T2T prepends a short mentor-generated guidance text to the student's prompt; C2C fuses the mentor's layer-wise KV cache into the student's cache once before decoding [Fu et al., 2026]; LoRA fine-tunes the student with the same trainable parameter count as the MentorPulse bridge [Hu et al., 2022]; MentorPulse (MP) is the full system of Section 4.

Metric and defaults. Recovery rate measures the fraction of the mentor–student gap closed: $\mathrm{Rec} = (\mathrm{score} - \mathrm{score}(\text{S-ONLY}))/(\mathrm{score}(\text{M-ONLY}) - \mathrm{score}(\text{S-ONLY})) \times 100\%$. Values are not truncated; macro averages include datasets whose anchor gap exceeds one standard error (true for all runs). Unless stated otherwise, MP uses $R{=}16$ (Sections 5.5 and 5.6); C2C, LoRA, and MentorPulse share the same training data and budget; all methods use greedy decoding with reasoning disabled, averaged over three seeds. Appendix D gives templates, trainable capacities, and search ranges.

5.2 Overall Evaluation

Figure 4 gives the per-dataset picture on A2, where the anchors sit 6.3 to 27.7 points apart. MP closes 52.2% of that gap on macro average, ahead of T2T at 26.9%, LoRA at 17.5%, and C2C at 10.9%. rT2T refreshes the guidance text every 16 tokens under a protocol matched to MP (Appendix C.2); it lifts macro recovery only to 33.7%, significant over T2T on just two triggering tasks (Appendix F.4), while paying 41–54$\times$ MP's per-request cost on the long-output representatives (Appendix H): refreshing in text space points the right way but carries little, and the step to 52.2% belongs to the latent interface. The ordering is not uniform: on the five selection datasets (A–C and I–J) the two latent methods are close (MP 58.6%, C2C 50.1%): outputs of a few tokens leave static guidance no time to go stale. On the seven registered long-generation tasks (Table 3) the picture splits. On the five open-ended sets (G, H, K–M), C2C lands 1.7 to 5.1 points below the student line, clearing twice the pooled standard deviation on four of the five (on MultiNews the deficit is direction-consistent but within noise; the red drops in Figure 4 mark the significant flips). On the two math sets (D, E) C2C keeps small positive gains, in line with the staleness account of Section 3: static guidance turns harmful where the target state evolves with the generated text, whereas a math problem statement keeps determining the target throughout the derivation. Averaged over the seven registered tasks, C2C recovery is -15.8% against MP's 54.7% (-34.5% versus 56.5% on the five flipped sets). LiveCodeBench is the exception: no method recovers more than 7% of the gap, matching the capability limit in Section 3.

Figure 5 re-plots the same scores in a shared length coordinate: x and y are each dataset's median input and output token counts ($\log_2$, centered on the median dataset), outputs measured from M-ONLY generations. In the left panel C2C fades and turns red as output length grows: every red point (G, H, K, L, M) lies in the upper half. In the right panel

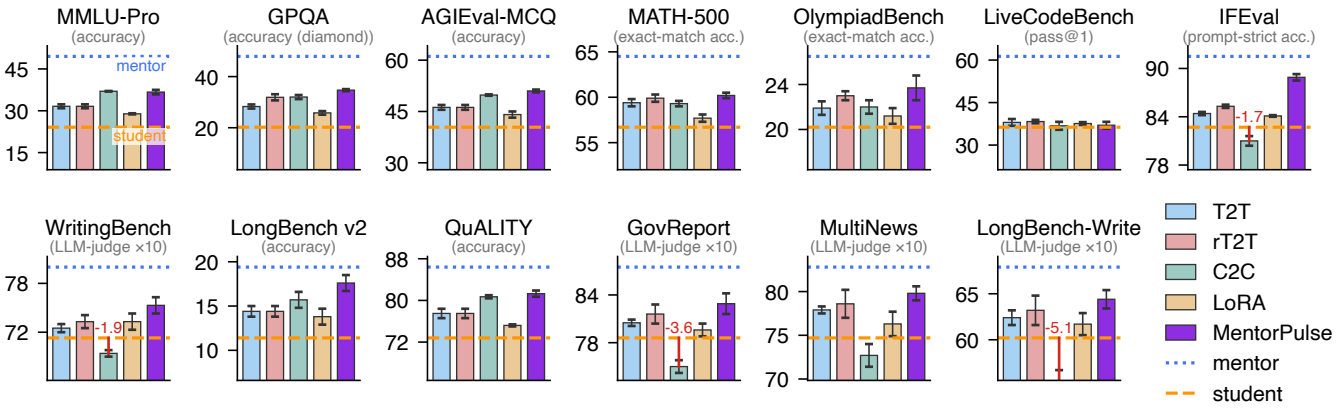

Figure 4: Results on the thirteen datasets for A2; MP uses R=16, and rT2T refreshes the T2T guidance text on the same schedule (Appendix C.2). Dotted/dashed lines are the M-ONLY/S-ONLY anchors; error bars are standard deviations over three seeds; each y-axis is clipped to its anchor range.

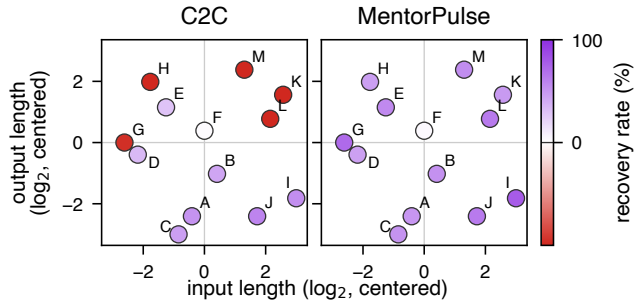

Figure 5: The thirteen datasets in one length coordinate system; color is each panel's recovery rate, clipped to $[-20, 100]$, red marking harm. Coordinates are spread order-preservingly for readability (distances not to scale); point positions are identical in both panels.

the upper half stays purple and the lower half matches C2C closely; short outputs rarely reach a refresh boundary, so MP operates there as a static bridge. The MP upper half remains slightly lighter than its lower half: refresh reduces but does not remove the difficulty of long generation. The code point F stays near white in both panels.

5.3 Ablations

Table 1 removes one design at a time. Removing refresh (row 2) is the largest mechanism-side drop: mean recovery on the two long-output sets falls from 59.4% to -24.4%, while the short-answer columns do not move a cell: those outputs end before the first refresh boundary. Row 3 separates schedule from training: swapping fresh memory into a bridge without windowed refresh training recovers little ($+1.2$ and $+2.2$ over the static row), and the training contributes the rest ($+8.3$ and $+3.1$); on IFEval both steps clear twice the pooled standard deviation, whereas on the LLM-judged GovReport they are direction-consistent but within noise, so the mechanism attribution reads from IFEval: the bridge must learn to digest mid-generation memory changes. Row 4 matches the full system: incremental refresh loses nothing against full recomputation, and Section 5.6 prices its cost advantage. The content controls repeat the attribution logic of Section 3: mismatched memory falls below the floor, random memory hovers near it, and zeroed gates reproduce S-ONLY bit for bit. The construction ablations (rows 8–10, 12) each cost several recovery points, tail slots mattering most on IFEval. Row 11 is designed not to drop: the top-16-layer read-out stays within noise on all four sets, making the pruned-layer transmission

Table 1: Ablations on A2 at R=16, on two long-output sets (G, K) and two short-answer controls (A, J); ΔRec. is the change in mean recovery over the four sets relative to row 1. Rows 4 and 11 should stay close to row 1, whereas row 7 must reproduce S-ONLY; standard deviations in Appendix F.

#	Ablation	G	K	A	J	ΔRec.
1	Full system (R=16)	88.9	82.9	36.7	81.3	—
Refresh mechanism						
2	No refresh ($R \to \infty$)	79.4	77.6	36.7	81.3	−41.9
3	Refresh w/o WRT	80.6	79.8	36.7	81.3	−32.3
4	Full re-prefill	88.6	83.0	36.7	81.3	−0.6
Memory content						
5	Mismatched mem.	79.0	77.5	23.8	72.5	−72.1
6	Random mem.	81.5	78.6	23.8	72.3	−62.3
7	Gates zeroed	82.7	78.6	24.1	72.8	−57.7
Memory construction						
8	No tail slots	85.1	81.6	36.0	80.3	−17.0
9	Mean-pooled slots	87.6	82.0	36.4	80.6	−7.8
10	Uniform read-out	88.2	82.0	36.2	80.7	−6.1
11	Top-16 layers	88.5	82.6	36.5	80.9	−2.9
Refresh details						
12	Two seg. types	87.9	81.4	36.7	81.1	−7.4

Table 2: Macro-average recovery (%) over the thirteen datasets by mentor group; best per row in bold. N_M/N_S is the parameter ratio; MP uses R=16; averages follow Section 5.1; standard deviations in Appendix F; model names in Figure 6.

Pair	N_M/N_S	T2T	C2C	LoRA	MP
A1 (27B→9B)	3.0	25.3	9.2	16.3	**54.0**
A2 (27B→4B)	6.8	26.9	10.9	17.5	**52.2**
A3 (27B→2B)	13.5	18.9	8.7	11.8	**35.7**
B1 (31B→12B)	2.6	18.8	7.1	16.9	**42.1**
C1 (32B→8B)	4.0	25.5	11.9	14.9	**54.6**
C2 (32B→4B)	8.2	22.4	11.4	16.0	**45.8**
C3 (32B→1.7B)	19.3	14.5	5.9	9.6	**26.8**
D1 (14B→8B)	1.8	28.6	12.3	15.6	**57.1**
D2 (14B→4B)	3.7	25.2	13.7	16.0	**42.7**
D3 (14B→1.7B)	8.7	14.8	8.5	11.0	**27.8**
E1 (14B→8B)	1.8	26.4	11.3	13.9	**53.0**

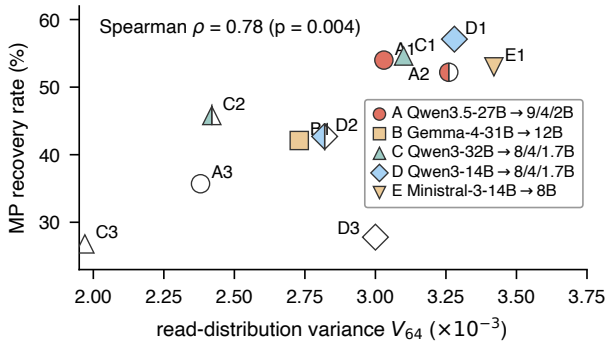

Figure 6: Read-distribution variance V_{64} against MP recovery (values as in Table 2); marker and color encode the mentor group. Only the rank correlation is reported.

of Section 5.6 viable.

5.4 Generality and Applicability

Table 2 shows the result survives a change of models: MP has the highest macro-average recovery on all eleven pairs and in all three families. The margins are smallest on C3 and D3, the most capability-separated pairs of their mentor groups: within each group, recovery falls monotonically as the parameter ratio grows (e.g., D1 57.1% to D3 27.8%).

During decoding, each bridge layer forms an attention distribution over the memory slots. V_{64} is the variance of this read distribution, averaged over heads, layers, the first 64 decoded steps, and 32 samples per dataset, computable in minutes of GPU time per pair. High variance means selective reading; low variance means attention spread almost uniformly, diluting the guidance. Recovery tracks V_{64} (Spearman $\rho = 0.78$, $p = 0.004$; Figure 6), and the weakest pair,

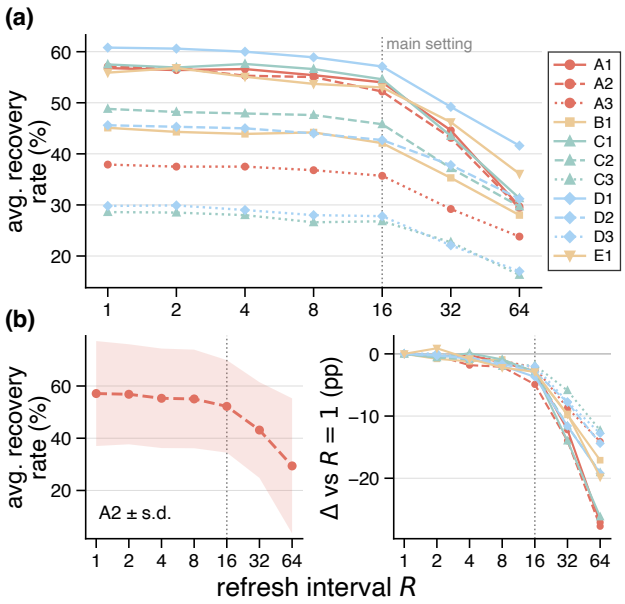

Figure 7: Refresh-interval sweep at inference. (a) Macro-average recovery for all eleven pairs (exclusion rule of Section 5.1); the dotted line marks R=16. (b) A2 with its cross-dataset band, and change relative to R=1 in points.

C3, has the lowest value—consistent with the reading that when the gap is too large the student cannot tell which slots matter, and reading degrades toward an average over the memory. Three limits apply: eleven pairs are observational evidence, V_{64} is comparable only under one training recipe, and the check screens pairs rather than replacing evaluation.

5.5 Refresh Interval

Figure 7 sweeps $R \in \{1, 2, 4, 8, 16, 32, 64\}$ at inference. All eleven curves share one shape: nearly flat for $R \leq 8$ (every pair within 2.2 recovery points of its R=1 value, small non-monotonic wobbles inside noise), declining from R=16, and down 12 to 28 recovery points by R=64; the shape is a property of the method: the plateau says guidance stays fresh for around ten tokens, so refreshing faster buys nothing. The main setting R=16 sits just past the plateau by intent: it gives up at most 4.9 recovery points against R=1, with long-output scores within one pooled standard deviation of R=8, and Section 5.6 places it among the cost-efficient settings $\{8, 16, 32\}$. R=8 stays among them but doubles the synchronizations, raising long-output per-request cost 35–50%; at least one long-output task loses cost dominance at $R \leq 4$. Quality-sensitive deployments with spare bandwidth can prefer R=8.

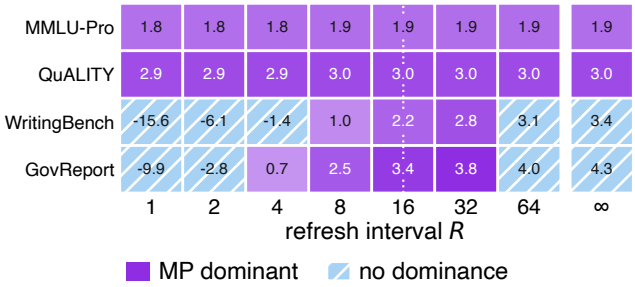

Figure 8: Cost–quality dominance against T2T on A2. Purple: MP dominates; hatched: neither. Cell numbers give the per-request saving $\mathrm{Cost}_{\mathrm{T2T}} - \mathrm{Cost}_{MP}$ in 10^{-3} dollars; the dotted line marks R=16. rT2T is omitted, cost-dominated at every evaluated R (Table 27).

5.6 Cost Accounting

We price with measured throughput, not theoretical FLOPs: prefill is compute-bound while decoding is bandwidth-bound, so FLOP counting understates decoding. Per-request cost sums each model's measured prefill and decode time at its GPU's rental rate, plus, for MP, the time to ship $\mathrm{Bytes}(R)$ (Eq. 1) per synchronization (formulas in Appendix H). Two structural facts follow: only the synchronization term grows as R shrinks, since mentor compute is R-independent (Section 4.3); and T2T pays for mentor decoding of the guidance text, the most expensive per-token operation, which MP never performs—rT2T pays it at every refresh and never reaches a dominance panel (Table 27). One R=16 refresh with 16 transmitted layers ships 2.7 MB in a measured 165 ms, versus $\sim$268 MB for a fused KV cache at a 2K-token input.

For each R and the static limit ∞, we compare (quality, cost) by dominance—at least as good on both, strictly better on one—with no weighted score. Figure 8 covers four representative tasks, one per length type, under blocking cross-site synchronization (billing and sensitivities in Appendix H). On the long-output tasks MP dominates T2T at the evaluated settings $R \in \{8, 16, 32\}$ (GovReport already at R=4), R=16 in the middle; smaller R keeps the highest quality but can lose cost dominance to synchronization overhead, and R=64 and ∞ are cheaper but no longer better. The short-output rows stay purple at every segment including ∞, with savings barely changing with R: refresh never triggers there, so the advantage is the static bridge's and we do not credit it to refresh. T2T-dominant segments never occur: text guidance always pays the mentor's decoding tax, so MP is strictly cheaper and can lose only on quality.

6 Conclusion

One-shot latent guidance does not stay reliable over long-form generation in our evaluated settings: static guidance falls below the student-only floor on long outputs, and a zero-training prefix-conditioned swap repairs it without touching the student's weights or cache. MentorPulse keeps guidance fresh through versioned slot memory, incremental mentor prefill, and windowed refresh training; across thirteen datasets and eleven pairs, refresh survives ablation as the main source of long-output gain and measured costs give the interval a defensible range. The gains require long outputs (short outputs never refresh; that advantage is the static bridge's), basic student capability, and a pair close enough for selective reading, screened by V_{64}. Guidance freshness is thus a design axis alongside capacity; learned triggers, prefill-free handoff, and cross-tokenizer pairs are future work.

References

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664, 2025a.

Yushi Bai, Jiajie Zhang, Xin Lv, Linzhi Zheng, Siqi Zhu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongWriter: Unleashing 10,000+ word generation from long context LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025b.

Benjamin Bergner, Andrii Skliar, Amelie Royer, Tijmen Blankevoort, Yuki Asano, and Babak Ehteshami Bejnordi. Think big, generate quick: LLM-to-SLM for fast autoregressive decoding. In *ICML Workshop on Efficient Systems for Foundation Models (ES-FoMo)*, 2024.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023.

Hanzhu Chen, Lin Yang, Jie Wang, Junhao Yan, Zhe Wang, Xize Liang, and Jianye Hao. Latent-guided reasoning: Empowering small LLMs with large-model thinking. In *International Conference on Learning Representations (ICLR)*, 2026.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024.

Yi Chen, JiaHao Zhao, and HaoHao Han. A survey on collaborative mechanisms between large and small language models, 2025.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3829–3846, 2023.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge, 2018.

Alexander Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir Radev. Multi-News: A large-scale multi-document summarization dataset and abstractive hierarchical model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1074–1084, 2019.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. In *International Conference on Learning Representations (ICLR)*, 2026.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*, 2024.

Chaoqun He, Renjie Luo, Yuzhuo Bai, Shengding Hu, Zhen Thai, Junhao Shen, Jinyi Hu, Xu Han, Yujie Huang, Yuxiang Zhang, Jie Liu, Lei Qi, Zhiyuan Liu, and Maosong Sun. OlympiadBench: A challenging benchmark for promoting AGI with olympiad-level bilingual multimodal scientific problems. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3828–3850, 2024a.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, et al. Multi-IF: Benchmarking LLMs on multi-turn and multilingual instructions following, 2024b.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, 2021.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Yunhai Hu, Zining Liu, Zhenyuan Dong, Tianfan Peng, Bradley McDanel, and Sai Qian Zhang. Speculative decoding and beyond: An in-depth survey of techniques, 2025.

Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1419–1436, 2021.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling bert for natural language understanding. In *Findings of the association for computational linguistics: EMNLP 2020*, pages 4163–4174, 2020.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, et al. Universal model routing for efficient llm inference, 2025.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 3045–3059, 2021.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pages 19274–19286. PMLR, 2023.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. EAGLE: Speculative sampling requires rethinking feature uncertainty. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 28935–28948. PMLR, 2024.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2024.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1773–1781, 2023.

Jesse Mu, Xiang Li, and Noah Goodman. Learning to compress prompts with gist tokens. *Advances in Neural Information Processing Systems*, 36:19327–19352, 2023.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, and Samuel R. Bowman. QuALITY: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358, 2022.

Pratyush Patel, Esha Choukse, Chaojie Zhang, Aashaka Shah, Íñigo Goiri, Saeed Maleki, and Ricardo Bianchini. Splitwise: Efficient generative LLM inference using phase splitting. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 118–132. IEEE, 2024.

Ruoyu Qin, Zheming Li, Weiran He, Jialei Cui, Feng Ren, Mingxing Zhang, Yongwei Wu, Weimin Zheng, and Xinran Xu. Mooncake: Trading more storage for less computation—a KVCache-Centric architecture for serving LLM chatbot. In *23rd USENIX Conference on File and Storage Technologies (FAST 25)*, pages 155–170. USENIX Association, 2025.

Haoran Que, Feiyu Duan, Liqun He, Yutao Mou, Wangchunshu Zhou, Jiaheng Liu, Wenge Rong, Zekun Moore Wang, Jian Yang, Ge Zhang, et al. HelloBench: Evaluating long text generation capabilities of large language models, 2024.

Qwen Team. Qwen3.5. https://qwen.ai/blog?id=qwen3.5, 2026. Accessed: 2026-07-28.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level Google-proof Q&A benchmark. In *First Conference on Language Modeling*, 2024.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Fali Wang, Jihai Chen, Shuhua Yang, Ali Al-Lawati, Linli Tang, Hui Liu, and Suhang Wang. A survey on collaborating small and large language models for performance, cost-effectiveness, cloud-edge privacy, and trustworthiness, 2025.

Yu Wang, Yifan Gao, Xiusi Chen, Haoming Jiang, Shiyang Li, Jingfeng Yang, Qingyu Yin, Zheng Li, Xian Li, Bing Yin, Jingbo Shang, and Julian McAuley. MemoryLLM: Towards self-updatable large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 50453–50466. PMLR, 2024a.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *Advances in Neural Information Processing Systems*, volume 37, 2024b.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with OSS-instruct. In *International Conference on Machine Learning (ICML)*, 2024.

Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, and Fei Huang. WritingBench: A comprehensive benchmark for generative writing. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Fangyuan Xu, Tanya Goyal, and Eunsol Choi. RefreshKV: Updating small KV cache during long-form generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24878–24893. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.acl-long.1211.

Haozhen Zhang, Tao Feng, and Jiaxuan You. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir Radev. QMSum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of NAACL-HLT*, 2021.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. AGIEval: A human-centric benchmark for evaluating foundation models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2299–2314, 2024a.

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 24)*, pages 193–210, 2024b.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. In *International Conference on Learning Representations (ICLR)*, 2025.

A Experimental Setup

A.1 Dataset Registry

Table 3 lists the thirteen evaluation datasets with the letter codes used throughout the paper, their metrics, and the evaluated subset sizes. LiveCodeBench (F) is restricted to problems released after July 2025, past the training cutoff of every evaluated model, as a contamination guard; QuALITY (J) uses the public dev split because test labels are not released. The long-generation suite (D, E, G, H, K, L, M) collects the generation-format tasks whose measured output medians span multiple refresh intervals (Appendix A.2); GPQA (B) is excluded as a selection task with only a brief justification (median 92), and LiveCodeBench (F), which qualifies by length (median 520), is excluded as the registered capability-limited exception (Section 5.2).

A.2 Length Profiles and the Short-Answer Protocol

Table 4 registers the length profile behind the task-length map of Figure 5. Input length is the median token count over evaluation inputs; output length is the median over M-ONLY generations, since the mentor defines the target behavior; L_g is the median T2T guidance length on the four datasets where the template was tuned. All medians are computed with the Qwen3.5 tokenizer; re-measuring with the Gemma-4 and Ministral 3 tokenizers changes medians by 3.1% and 3.8% respectively and moves no dataset across a length bin.

Short-answer protocol. The four pure multiple-choice sets (A, C, I, J) are evaluated under a short-answer protocol: in no-think mode the student is asked to output only the option letter, the standard treatment of choice tasks in lm-eval-style harnesses. Their output medians are 5–7 tokens, so at the main setting R=16 these datasets literally never reach a refresh boundary: MP@16 operates as the static bridge on them. This single registered fact grounds three places in the paper: the "=static" cells of the R sweep (Table 15), the unchanging short-output rows of Figure 8, and the applicability statement of Section 5.4. GPQA (B) keeps a brief free-form justification (median 92 tokens) and serves as the transitional sample: refresh triggers a few times with near-zero benefit. MATH-500 (D) and OlympiadBench (E) span many refresh windows and are discussed with the long-output tasks.

A.3 Runtime Environment and Availability

Bridge checkpoints are named `fb6-<pair>-r16-e2` (eleven, one per pair), C2C fusers `c2c-fuser-<pair>-v1`, and LoRA adapters `lora-r64-<pair>`. Scoring uses lm-eval-harness 0.4.9 for the standard sets and our long-form evaluator longeval v2.1 for the judged sets (H, K, L, M and the judged diagnostic tasks), with GPT-4.1 as the fixed judge at temperature 0. All decoding is greedy with reasoning disabled (no-think), with per-dataset generation caps; seeds are $\{13, 42, 2026\}$. The three trainable objects share the training set fuse_v3 (51,908 rows; Appendix D). Results were collected between June 2 and July 26, 2026. Every reported score, including the M-ONLY and S-ONLY anchors, follows the protocol of Section 5.1: at least three seeds (Appendix A.4), the full registered evaluation subsets of Table 3, and the deployment serving stack (vLLM 0.19.1 in bf16). One backend exception is registered: the Gemma-4 family (pair B1) is not supported by this vLLM build and is served through HuggingFace transformers in bf16 with identical decoding settings. Within every pair, anchors and all compared methods run on the same backend, so no comparison in the paper mixes serving stacks; the cost model prices only the A2 deployment pair and is unaffected. Separately, a lightweight screening pass over all thirteen models and datasets (July 26, 2026; `limit`=500, single run, same serving stack, Gemma-4 through HF) sanity-checked the capability ordering assumed in pair selection; no number from that pass enters any table, and recovery rates in particular divide method scores and anchor scores drawn from the same full-subset runs. Serving throughputs for the cost model are measured separately on the deployment stack (Appendix H). Code, evaluation configurations, the data pipeline, per-seed outputs, and the scripts that derive every table in this appendix will be released.

A.4 Statistical Discipline

Every reported number is the mean over at least three seeds, written as mean±standard deviation; methods that train vary the training seed, inference-only conditions vary the evaluation seed. When two conditions a, b are compared, the pooled standard deviation is $\sigma_{ab} = \sqrt{\sigma_a^2 + \sigma_b^2}$, and any difference below $2\sigma_{ab}$ is described as "comparable" rather than better or worse; every claim of significance in the paper passes this threshold. Recovery is $(\text{score} - \text{S-ONLY})/(\text{M-ONLY} - \text{S-ONLY}) \times 100\%$, reported without truncation (negative and $>100\%$ values kept; only figure color scales clip, as stated in their captions). Datasets whose mentor–student anchor gap is below one standard error would be excluded from macro-average recovery as unstable; no evaluated pair triggered this rule.

Table 3: The thirteen evaluation datasets. Letter codes are fixed throughout the paper; the last column marks the long-generation suite (inclusion rule in the text; length profiles in Table 4).

Code	Dataset	Metric	Eval. subset	Long-gen. suite
A	MMLU-Pro	accuracy	2,000 (stratified)	—
B	GPQA	accuracy (diamond)	198	—
C	AGIEval-MCQ	accuracy	2,000 (stratified)	—
D	MATH-500	exact-match accuracy	500	✓
E	OlympiadBench	exact-match accuracy (OE-TO-maths-en)	675	✓
F	LiveCodeBench	pass@1	400 (post 2025-07)	—
G	IFEval	prompt-level strict accuracy	541	✓
H	WritingBench	LLM-judge mean $\times 10$	1,000	✓
I	LongBench v2	accuracy	503	—
J	QuALITY	accuracy	2,086 (dev)	—
K	GovReport	LLM-judge mean $\times 10$	200	✓
L	MultiNews	LLM-judge mean $\times 10$	200	✓
M	LongBench-Write	LLM-judge mean $\times 10$	120	✓

Table 4: Length profiles (median token counts). Outputs are measured from M-ONLY generations; the tokenizer is Qwen3.5's (cross-family deviation $<4\%$).

Code	Dataset	Input med.	Output med.	L_g med.
A	MMLU-Pro	410	6	130
B	GPQA	540	92	—
C	AGIEval-MCQ	320	5	—
D	MATH-500	140	430	—
E	OlympiadBench	280	570	—
F	LiveCodeBench	470	520	—
G	IFEval	110	486	—
H	WritingBench	160	940	250
I	LongBench v2	9,400	7	—
J	QuALITY	5,100	6	205
K	GovReport	7,600	705	310
L	MultiNews	6,300	566	—
M	LongBench-Write	4,300	1,080	—

B Condition Definitions and Paradigm Properties

B.1 Experimental Conditions

Table 5 defines every condition used in the paper. Main-text performance results use S-ONLY, M-ONLY, T2T, C2C, LoRA, and MP@16; the diagnosis section (Section 3) uses ONESHOT, ORACLESWAP@16, and the two mismatch controls; rT2T appears beside T2T in Figure 4 and in Appendices F.4 and H; the remaining rows are ablation controls (Table 1).

B.2 Paradigm Properties

Table 6 compares the closest cross-model interfaces and lossless speculative decoding on the three properties discussed in Section 2: whether the mentor stays frozen, whether the transferred state count is explicitly capped, and whether the transferred signal is refreshed from the generated prefix. Speculative decoding is included as a discussion-only contrast: it is lossless and couples the two models token by token on one machine, so quality and cost are not comparable with the lossy, sparsely-synchronized interfaces above it.

C Baseline Implementations and Fairness

Each baseline below states its implementation and the hyperparameter search it received, so that no comparison rests on an under-tuned opponent.

C.1 T2T

The guidance template `t2t-guide-v3` instructs the mentor to read the input and produce, in order, a task analysis, the key points a correct answer must satisfy, and a short answer plan; the guidance is truncated at $L_g \leq 384$ tokens and prepended to the student prompt. The template went through three iterations on held-out development data (v1: free-form summary; v2: added the key-point list; v3: added the answer plan and tightened the length cap), and v3 was selected by development-set student score.

C.2 rT2T

rT2T is the refreshed version of T2T: it fills the "text $\times$ refresh" cell of the $\{\text{text}, \text{latent}\} \times \{\text{static}, \text{refreshed}\}$ matrix,

Table 5: All experimental conditions. "Mentor decodes" refers to inference time only; offline label generation is excluded. ONESHOT and the ablation row MP−refresh are the same condition under two names (static bridge, $R{\to}\infty$).

Condition	Definition	Role	Mentor decodes
S-ONLY	student alone, no bridge, no memory	floor	—
M-ONLY	mentor answers alone	ceiling	yes
T2T	one-shot mentor guidance text ($L_g \leq 384$) prepended to the student prompt	text-interface route	yes (once)
rT2T@R	T2T guide refreshed every R tokens; replace-style rebuild, delimiter-stripped	text × refresh cell (Appendix C.2)	yes (every refresh)
C2C	mentor layer-wise KV fused into the student cache once before decoding	static-latent route	no
LoRA	equal-budget low-rank finetuning, no memory	parameter route	—
MP@R	full method after WRT, refresh interval R	main method (R=16)	no
MP-ST	MP with a learned staleness trigger	stage-2 outlook (Appendix I.3)	no
ONESHOT	bridge with static memory ($R{\to}\infty$; = MP−refresh)	one-shot guidance instance	no
ORACLESWAP@R	zero-training rebuild of the memory from $x \oplus \hat{y}_{<t}$, swapped every R tokens	staleness diagnosis	no
ORACLESWAP-MIS@R	initial memory matched; refresh points swap in *fresh* memory of another sample	refresh × content control	no
ONESHOT-mismatch	initial memory taken from another sample, never refreshed	static content canary	no
Random	random-vector memory of the correct shape	information control	no
Zero / gate-off	zero memory or closed gates	degeneracy check (= S-ONLY, bit-exact)	no

Table 6: Property comparison with the closest cross-model interfaces and speculative decoding. Refresh means updating transferred guidance from the generated prefix; dashes mark properties that do not apply.

Property	L2S	C2C	LGR	SPECDEC	**MP**
Mentor frozen	✓	✓	✗	✓	✓
Explicit state-count cap	✗	✗	✓	—	✓
Prefix-conditioned refresh	✗	✗	✗	—	✓
Matches mentor distribution	✗	✗	✗	✓	✗

so that the refresh schedule and the representation format are separated as factors. Its protocol is fixed by five clauses, each of which protects a specific fairness property:

1. **Mentor conditioning.** At each refresh the mentor incrementally prefills $[x; \hat{y}_{<t}]$, the same fresh path MentorPulse uses, and decodes a new hint of L'_g tokens (template `t2t-hint-v1`).

2. **Delimiters.** Hints are injected wrapped in `<guide>...</guide>` and stripped, delimiters and content, before scoring, so no hint text can leak into the graded output.

3. **Replace-style injection (main setting).** Each refresh discards the previous hint (including the initial guide) and rebuilds $[\text{hint}_j; x; \hat{y}_{<t}]$ with a full re-prefill. Replacement is strictly more expensive than accumulation in every (task, R, L'_g) cell of Table 27—by a second-order margin, about 2% of total cost—and keeps the context clean, mirroring MentorPulse's replacement of stale memory; we adopt the more expensive, cleaner variant as the most generous implementation. The accumulation variant is reported as a robustness check in Appendix F.4.

4. **Hint budget.** $L'_g \in \{64, 128\}$ is swept and the per-task best is reported (best-effort baseline).

5. **Initial guide = T2T.** This yields two identity checks for free: on short-answer tasks (output $<R$, n_{sync}=0) and at $R{\to}\infty$, rT2T must equal T2T cell by cell. Both identities are hard-verified by the fill scripts (Appendix G.5).

rT2T is the rigorous return of the retired TH-k condition of an earlier design: TH-k appended hints on a fixed schedule without conditioning the mentor on $\hat{y}_{<t}$, without replacement, and without delimiter stripping, and is superseded in all experiments.

C.3 C2C

The C2C reproduction follows the layer-wise cache-fusion design of Fu et al. [2026]: the mentor's per-layer KV cache for the input is projected and fused into the student's cache by a trainable fuser before decoding, and never updated afterwards. The fuser (`c2c-fuser-<pair>-v1`) is trained on the same data and budget as the bridge; its width is chosen per pair so that its trainable parameter count matches the bridge's, and its learning rate is searched over the same grid as MentorPulse's.

C.4 LoRA

The LoRA control uses rank 64 on all pairs, with adapters on the attention and MLP projections. Trainable parameter counts are matched to the bridge pair by pair, ranging from

21M to 58M across the eleven pairs, so any gain of MentorPulse over LoRA cannot be attributed to extra capacity. The learning rate is searched over the shared grid; training data and budget are identical to the bridge's.

C.5 Fairness Summary

The three trainable objects (C2C fuser, LoRA adapters, MentorPulse bridge) use the same training set fuse_v3, the same step budget, and the same learning-rate search; all methods are served with the same stack, decode greedily without thinking mode, and are scored by the same per-dataset script; every number averages at least three seeds (Appendix A.4). rT2T contains no trainable object; its fairness rests on the five-clause protocol above.

D Training Details

D.1 Training Data

Two offline passes prepare the training store. The first pass generates long targets: prompts are drawn from open instruction sources for writing, explanation, and rewriting, code instructions in the OSS-Instruct style [Wei et al., 2024], synthetic instruction-following prompts with verifiable constraints [Zhou et al., 2023], and long document-grounded targets. The mentor decodes greedily without thinking mode, up to 768 new tokens; this is the only mentor decoding in the entire pipeline and happens once, offline. Four quality gates screen the generations: a repetition gate (4-gram), a completeness gate (natural EOS within a length window), a language gate, and a code gate rejecting broken fenced blocks. Each gate removes 5–15% of the rows it screens; the failures overlap, and together the four gates drop 24.4% of prompts. For the 27B mentor, 26,997 prompts yield 20,408 long labels (75.6% kept). The second pass extracts mentor states: prefill only, all layers, written under the slot layout to a store of about 1.9 TB in bf16. The mixed set fuse_v3 combines all 20,408 filtered long targets (39.3%) with 31,500 short-answer rows downsampled from the static stage (60.7%), 51,908 rows in total; multiple-choice rows restrict the answer alphabet to the true option count of each row to remove a leakage artifact. All sources are disjoint from the thirteen evaluation sets and from the diagnostic suite (Appendix G.1), so no refresh gain can come from memorized evaluation data.

D.2 Windowed Refresh Training

WRT samples a refresh point t per row (at the $R{=}16$ grid, including $t{=}0$, which keeps static training inside the objective) and computes the loss only on the window $(t, t{+}R]$; the prefix enters as the source of the refreshed memory, never as supervision. By causality, one extraction pass over prompt-plus-label provides the mentor states for every candidate refresh time, so storage grows only linearly (about 768 extra tokens per row). An optional second round replaces a subset of label prefixes with prefixes sampled from the bridged student, following dataset aggregation [Ross et al., 2011], to shrink the train–test prefix mismatch stated in Section 4.

D.3 Optimization and Budget

Both backbones, embeddings, and heads are frozen; the trained parameters are the shared projector, the per-layer rank-64 corrections, the layer, position, and segment embeddings, the read gates ρ_o, and the per-layer cross-attention blocks (rank 256). Training runs 2 epochs at learning rate 2×10^{-5} (selected on the development split from the grid shared with C2C and LoRA) with answer-segment cross-entropy, weighted by token share. The bridge warm-starts from the static-stage checkpoint with gates not reset; inherited positions map to the tail segment of the extended position table and all new entries initialize at zero, so the extended system is bit-identical to the inherited one at step zero. Microbatches are grouped by a slot budget of 256 rather than a row count, because the injected memory is projected to K/V at every injection layer and retained for backward; gradient checkpointing is disabled as incompatible with the forward hooks that mount the bridge. Generation health during development is monitored with distinct-2, 4-gram repetition, natural-EOS rate, and mean length; alarms fire below 90% EOS rate or above 10% repetition. Table 7 consolidates the pipeline statistics.

Table 7: Training pipeline and hyperparameter summary.

Item	Value
Long-target prompts / kept labels	26,997 / 20,408
Quality-gate filter rate	5–15% per gate, 24.4% total
Mixed training rows (fuse_v3)	51,908
Long / short-answer mix	20,408 / 31,500 (39% / 61%)
Mentor state store (27B, bf16)	$\sim$1.9 TB
Max label length	768 tokens
Learning rate / epochs	2×10^{-5} / 2
Bridge rank / per-layer correction rank	256 / 64
Slot budget per microbatch	256
WRT refresh-point grid	$R{=}16$, incl. $t{=}0$
EOS-rate alarm / repetition alarm	$<$90% / $>$10%

E Method Implementation Details

E.1 Slot Memory Layout

For each selected mentor layer o, retained states are normalized by a fixed scalar rms_o and mapped from d_m to d_t by a shared projection with a rank-64 layer-specific correction.

Segment slots summarize the prompt and the generated prefix, each taking the true hidden state at its segment's final position; the current tail keeps 32 raw-state slots for exact recent constraints. Prompt segments default to $S{=}32$ tokens under the cap $p_x \leq P{=}128$: longer inputs are re-segmented into P evenly coarser segments (roughly 73-token segments at the 9,400-token LongBench v2 median input), which caps the initial transfer regardless of input length (Appendix H). Generated-prefix slots are appended at one slot per 32 tokens under a separate budget of 128 slots (4,096 output tokens) with oldest-first eviction as a backstop; prompt-summary slots are never evicted. No per-dataset generation cap (Appendix A.3) reaches this budget — the longest registered output median, 1,080 tokens on LongBench-Write, fills 34 slots (Table 4) — so the backstop never fires in our runs. Three segment-type embeddings distinguish prompt summary, generated-prefix summary, and current tail; layer and slot-order embeddings complete the layout, and a validity mask handles variable lengths. A learned weight ρ_o softly combines mentor layers and defines the pruning order used in Section 5.6. All evaluated pairs share a tokenizer family, so the mentor consumes student-generated tokens directly.

E.2 Gated Cross-Attention Bridge

Each student layer carries a gated cross-attention block whose query, key, value, and output projections have rank 256; the query is the student residual stream and the memory supplies keys and values, projected locally on the student side (the wire payload is one slot tensor, not separate K/V tensors). Scalar gates start at zero, so the untrained bridge is an identity map and the system reproduces the plain student bit for bit — the degeneracy check used in Sections 3 and 5.3 and the gate-off row of Table 5. Training escapes this closed-gate dead zone by warm-starting from the static-stage checkpoint whose gates are already open, without resetting them; new position and segment entries initialize at zero so the inherited path is unchanged at the start of WRT.

E.3 Versioned Incremental Refresh

At a refresh the mentor incrementally prefills only the R newly generated tokens, appends the resulting segment slots, and rebuilds the tail; the memory tensor is replaced in place (hot swap) as a new version Z_{t_j}. Because the memory is a side tensor rather than part of the student's self-attention cache, the student's KV cache and generated text are never invalidated; older summary slots are retained on the student side and are not retransmitted, so each refresh ships only the new summary slots and the current tail. Row 4 of Table 1 verifies that this incremental path matches full recomputation within noise.

F Complete Experimental Results

F.1 Main Pair

Table 8 gives the full per-dataset scores with standard deviations for the main pair A2, the source of Figure 4; Table 9 gives the raw scores behind the recovery-based ablation Table 1. On the sign-flip claim, C2C sits below S-ONLY on five of the seven registered long-generation tasks (G -1.7, H -1.9, K -3.6, M -5.1, each clearing twice the pooled standard deviation, thresholds 1.13–4.25; on MultiNews the deficit of -2.0 against a 2.86 threshold is direction-consistent but within noise, and the main text treats that set accordingly), and keeps small positive gains on the two math sets (D $+2.6$, E $+1.8$). No short-output or capability-limited set flips. In Table 9, the A and J columns of rows 2–4 equal row 1 cell by cell: those outputs end before the first refresh boundary, so every initially-matched refresh condition is structurally identical to the full system there (the zero-trigger identity of Appendix G.5); row 5 drops on A and J because its *initial* memory is already mismatched, a static-mismatch condition.

F.2 All Eleven Pairs

Tables 10–14 give the complete per-dataset results for the ten remaining pairs; A2 is in Table 8. The macro-recovery rows reproduce Table 2 and the $R{=}16$ column of Table 16. Capability-limited registration: on B1, the Gemma-4-12B student scores near zero on MATH-500 and OlympiadBench, far below these tasks' difficulty floor. Consistent with the capability limit of Section 3, no comparison method recovers a meaningful fraction of such a gap, so the derived columns follow a capability-limited profile (near-zero gains for every method) and these two cells are excluded from the long-output gain narrative.

F.3 Refresh-Interval Sweep

Table 15 gives the per-dataset A2 scores behind Figure 7 and Table 16 the macro-average recovery of all pairs. Every R reuses the same bridge checkpoint (`fb6-A2-r16-e2`, trained with refresh points sampled at $R{=}16$); the sweep changes only the inference-time interval. A train–test interval mismatch would show as an anomaly at the matched column $R{=}16$ relative to its neighbors; the curves are smooth through it, so no per-R retraining was triggered. Starred cells are the short-answer identities of Appendix A.2: from $R{=}8$ on, A, C, I, and J trigger zero refreshes and equal the static bridge exactly (single registered source, shared with row 2 of Table 9); at $R \leq 4$ they trigger one to six times with effects inside noise. The static-bridge value of G (79.4) does not appear in this table because G still refreshes multiple times at $R{=}64$ (output median 486).

Table 8: Full results for the main pair A2 (Qwen3.5-27B → Qwen3.5-4B), mean±sd over three seeds; MP@16 throughout. The macro-recovery row uses the definition of Appendix A.4.

Code	Dataset	M-ONLY	S-ONLY	T2T	C2C	LoRA	MP@16
A	MMLU-Pro	49.5±0.1	24.1±0.2	31.6±0.7	37.0±0.1	28.9±0.2	36.7±0.8
B	GPQA	47.9±0.2	20.2±0.2	28.3±0.8	32.0±0.8	25.8±0.7	34.7±0.4
C	AGIEval-MCQ	61.1±0.1	40.4±0.0	46.2±0.7	49.8±0.2	44.1±0.9	51.0±0.4
D	MATH-500	64.5±0.1	56.7±0.1	59.4±0.4	59.3±0.3	57.7±0.4	60.2±0.3
E	OlympiadBench	26.5±0.1	20.2±0.1	21.9±0.6	22.0±0.6	21.2±0.7	23.7±1.1
F	LiveCodeBench	61.3±0.2	36.3±0.2	38.0±1.2	36.8±1.4	37.6±0.5	37.0±1.2
G	IFEval	91.5±0.0	82.7±0.0	84.4±0.2	81.0±0.6	84.1±0.1	88.9±0.4
H	WritingBench	80.0±0.5	71.3±0.4	72.5±0.5	69.4±0.4	73.3±1.0	75.3±1.0
I	LongBench v2	19.4±0.0	11.4±0.1	14.4±0.6	15.7±0.9	13.8±0.9	17.6±0.9
J	QuALITY	86.4±0.1	72.8±0.1	77.5±0.9	80.7±0.3	75.2±0.2	81.3±0.6
K	GovReport	87.5±0.5	78.6±1.4	80.5±0.4	75.0±0.8	79.6±0.8	82.9±1.3
L	MultiNews	82.8±0.4	74.7±0.6	77.9±0.4	72.7±1.3	76.3±1.4	79.8±0.8
M	LongBench-Write	67.9±0.6	60.2±1.4	62.4±0.8	55.1±1.6	61.7±1.2	64.4±1.0
Macro recovery (%)		—	—	26.9	10.9	17.5	52.2

Table 9: Raw scores with standard deviations for the twelve ablation rows of Table 1 (A2, R=16; two long-output sets G, K and two short-answer controls A, J). "Reused" rows are copied from their registered source, not re-evaluated.

#	Condition	G	K	A	J	Runs
1	MP (full system, R=16)	88.9±0.4	82.9±1.3	36.7±0.8	81.3±0.6	reused (= MP@16, Tab. 8)
Refresh mechanism						
2	MP−refresh ($R \to \infty$)	79.4±0.2	77.6±0.9	36.7±0.8	81.3±0.6	retrained
3	MP−WRT (zero-training swap)	80.6±0.5	79.8±0.9	36.7±0.8	81.3±0.6	inference only
4	MP−full (full re-prefill)	88.6±0.2	83.0±0.9	36.7±0.8	81.3±0.6	inference only
Memory content						
5	MP−mismatch (incl. initial)	79.0±0.4	77.5±0.7	23.8±0.5	72.5±0.8	inference only
6	MP−random	81.5±0.8	78.6±1.1	23.8±0.7	72.3±0.5	inference only
7	MP−gate0	82.7±0.0	78.6±1.4	24.1±0.2	72.8±0.1	reused (= S-ONLY); bit-identity passed
Memory construction						
8	MP−tail	85.1±0.2	81.6±1.2	36.0±0.2	80.3±0.3	retrained
9	MP−meanpool	87.6±0.3	82.0±1.0	36.4±0.5	80.6±0.8	retrained
10	MP−uniform	88.2±0.3	82.0±0.8	36.2±0.4	80.7±0.2	retrained
11	MP−prune16 (top-16 layers)	88.5±0.7	82.6±0.6	36.5±0.9	80.9±0.6	inference only
Refresh engineering						
12	MP−2seg	87.9±0.2	81.4±0.6	36.7±0.5	81.1±0.6	retrained

Table 10: Full per-dataset results, pairs A1 and A3 (datasets by letter code, Table 3); mean±sd over three seeds.

	M-ONLY	S-ONLY	T2T	C2C	LoRA	MP@16
A1: Qwen3.5-27B → Qwen3.5-9B ($N_M/N_S = 3.0$)						
A	49.5±0.1	27.4±0.1	34.7±0.6	39.0±0.3	31.4±0.2	39.6±0.2
B	47.9±0.2	27.9±0.0	33.1±0.7	38.3±0.6	31.9±0.9	40.0±0.9
C	61.1±0.1	47.4±0.1	52.1±0.5	54.1±0.5	50.1±0.4	55.2±0.4
D	64.5±0.1	58.3±0.0	59.4±0.8	59.6±0.3	59.4±0.9	60.9±0.9
E	26.5±0.1	21.7±0.0	23.1±0.8	22.5±1.2	23.0±0.7	24.5±1.4
F	61.3±0.2	49.7±0.2	50.1±0.3	49.5±0.4	50.2±0.8	50.4±0.6
G	91.5±0.0	85.1±0.2	86.1±0.3	82.4±0.7	85.8±0.4	89.1±0.3
H	80.0±0.5	72.6±0.6	74.3±1.1	70.1±1.2	73.2±0.6	76.7±0.7
I	19.4±0.0	14.7±0.1	15.8±0.4	16.7±0.3	14.9±0.3	18.1±0.8
J	86.4±0.1	76.4±0.1	79.9±0.5	81.9±0.3	78.1±0.5	83.0±0.2
K	87.5±0.5	81.6±0.6	83.0±0.4	79.8±0.7	82.8±1.2	85.6±0.6
L	82.8±0.4	77.4±0.4	79.3±0.2	75.4±0.6	78.9±0.7	79.5±0.5
M	67.9±0.6	61.1±0.9	63.1±0.6	59.5±0.9	62.2±1.3	65.2±1.0
Macro rec. (%)		—	25.3	9.2	16.3	54.0

	M-ONLY	S-ONLY	T2T	C2C	LoRA	MP@16
A3: Qwen3.5-27B → Qwen3.5-2B ($N_M/N_S = 13.5$)						
A	49.5±0.1	16.0±0.0	23.3±0.3	27.3±0.5	20.5±0.4	28.3±0.2
B	47.9±0.2	19.9±0.1	25.6±1.3	29.6±1.2	23.0±0.8	30.8±0.3
C	61.1±0.1	32.6±0.0	38.0±0.9	42.0±0.1	35.3±0.5	42.5±0.8
D	64.5±0.1	51.4±0.0	53.8±0.5	54.3±0.4	52.9±0.6	56.1±0.6
E	26.5±0.1	14.6±0.2	16.7±0.8	16.2±0.5	16.4±0.3	19.4±1.3
F	61.3±0.2	13.8±0.2	16.6±1.6	13.9±0.3	16.1±0.3	15.4±0.5
G	91.5±0.0	72.4±0.1	74.9±0.4	68.8±0.4	74.1±0.2	81.6±0.3
H	80.0±0.5	67.9±0.4	69.6±0.3	65.6±0.4	69.0±0.4	71.6±0.4
I	19.4±0.0	10.5±0.1	13.3±0.3	13.5±0.3	12.1±0.3	13.9±1.1
J	86.4±0.1	54.2±0.2	61.0±0.3	66.0±0.4	57.4±0.5	68.6±0.3
K	87.5±0.5	74.5±0.5	77.3±0.5	72.4±0.4	76.8±0.5	79.5±1.8
L	82.8±0.4	69.1±1.3	72.4±1.2	66.2±1.4	70.0±0.3	74.2±1.1
M	67.9±0.6	55.1±0.9	57.3±1.0	52.6±0.5	57.3±1.7	59.8±1.3
Macro rec. (%)		—	18.9	8.7	11.8	35.7

Table 11: Full per-dataset results, pairs B1 and C1 (see the B1 capability-limited note in Appendix F.2).

	M-ONLY	S-ONLY	T2T	C2C	LoRA	MP@16
B1: Gemma-4-31B → Gemma-4-12B ($N_M/N_S = 2.6$)						
A	65.6±0.1	44.3±0.0	50.4±0.4	54.0±0.4	48.2±0.3	54.3±0.3
B	51.5±0.2	42.4±0.0	45.4±1.0	46.4±0.4	44.5±0.8	47.5±0.8
C	68.0±0.0	51.3±0.2	56.3±0.1	58.4±0.1	54.2±0.5	59.3±0.7
D	54.2±0.0	10.1±0.2	11.9±0.9	10.6±0.6	13.1±0.2	12.2±1.2
E	31.5±0.1	2.1±0.0	3.9±0.3	1.9±0.3	3.8±0.4	3.1±0.7
F	67.8±0.2	62.4±0.0	62.7±0.5	62.5±0.6	62.8±0.7	62.5±0.4
G	93.6±0.1	87.7±0.1	88.0±0.8	85.8±0.3	88.6±0.3	92.4±0.2
H	79.6±0.4	72.7±0.9	74.1±0.5	70.7±0.3	74.0±0.2	75.9±1.1
I	8.6±0.0	5.2±0.1	5.8±0.5	7.0±1.0	7.3±0.8	6.7±1.4
J	88.1±0.0	81.3±0.1	83.2±0.3	85.7±0.7	82.8±0.7	85.5±0.7
K	78.9±1.3	74.5±0.6	76.0±0.6	73.5±1.2	75.0±0.7	76.3±1.0
L	81.0±0.4	74.9±0.8	76.5±0.7	73.2±0.4	76.3±0.3	78.4±0.3
M	65.6±0.8	60.4±0.7	60.7±1.1	57.9±0.5	59.8±1.4	63.3±1.1
Macro rec. (%)		—	18.8	7.1	16.9	42.1

	M-ONLY	S-ONLY	T2T	C2C	LoRA	MP@16
C1: Qwen3-32B → Qwen3-8B ($N_M/N_S = 4.0$)						
A	20.4±0.0	6.1±0.1	10.8±0.3	13.3±0.2	8.7±0.2	13.3±0.6
B	24.9±0.1	13.5±0.2	16.9±1.0	19.1±1.2	16.7±0.8	19.5±0.7
C	25.1±0.1	12.0±0.0	16.1±0.3	18.2±0.8	14.5±0.9	19.2±0.2
D	69.7±0.2	63.5±0.0	65.3±0.2	65.1±0.4	64.4±1.2	66.9±0.4
E	34.0±0.0	28.6±0.1	29.6±0.7	29.7±0.3	29.7±1.1	30.9±0.8
F	47.3±0.1	35.9±0.1	36.7±0.8	36.1±0.8	36.6±0.2	36.7±1.3
G	89.8±0.0	82.8±0.1	84.5±0.2	80.6±0.4	83.8±0.4	88.4±0.5
H	78.3±1.1	70.3±0.9	72.6±1.2	67.1±1.0	71.3±0.7	74.5±0.3
I	17.6±0.1	12.1±0.1	13.9±1.0	14.7±0.7	12.9±0.8	15.2±0.5
J	76.0±0.1	67.1±0.1	69.8±0.3	71.7±0.4	68.4±0.7	72.8±0.6
K	82.7±0.7	76.7±0.9	78.4±0.4	75.5±1.4	77.1±0.7	80.7±0.5
L	81.3±0.8	73.5±0.6	75.4±0.4	72.1±1.0	74.0±0.7	78.1±0.8
M	65.4±1.5	57.9±0.9	59.0±0.7	55.7±0.8	59.3±0.9	63.1±0.3
Macro rec. (%)		—	25.5	11.9	14.9	54.6

Table 12: Full per-dataset results, pairs C2 and C3.

	M-ONLY	S-ONLY	T2T	C2C	LoRA	MP@16
C2: Qwen3-32B → Qwen3-4B ($N_M/N_S = 8.2$)						
A	20.4±0.0	13.0±0.2	14.6±0.4	16.1±0.1	14.1±0.3	15.9±0.5
B	24.9±0.1	17.8±0.0	20.3±1.7	20.4±0.6	18.3±0.6	21.7±0.8
C	25.1±0.1	14.7±0.0	17.4±0.3	19.2±0.9	16.6±0.1	19.7±0.8
D	69.7±0.2	60.1±0.1	62.5±0.8	63.0±0.9	62.2±0.7	64.3±0.3
E	34.0±0.0	24.8±0.1	27.2±1.1	26.4±0.7	25.8±1.2	29.0±1.5
F	47.3±0.1	31.6±0.1	32.5±1.1	31.5±0.5	33.3±0.2	32.4±1.5
G	89.8±0.0	79.6±0.1	80.7±0.3	77.0±0.1	80.4±0.6	86.1±0.1
H	78.3±1.1	68.4±0.4	69.5±0.7	66.4±1.7	70.0±1.0	72.9±0.7
I	17.6±0.1	10.6±0.2	12.5±0.5	14.1±1.6	11.5±0.8	14.2±0.6
J	76.0±0.1	59.3±0.1	64.0±0.3	66.8±0.5	62.2±0.3	68.1±0.7
K	82.7±0.7	73.4±1.2	75.4±1.0	71.6±1.0	75.3±0.7	77.9±0.4
L	81.3±0.8	70.8±1.3	73.4±0.6	68.2±0.3	72.9±0.4	76.5±0.6
M	65.4±1.5	55.8±0.6	58.5±0.5	53.3±0.8	58.6±0.9	59.9±0.5
Macro rec. (%)		—	22.4	11.4	16.0	45.8

	M-ONLY	S-ONLY	T2T	C2C	LoRA	MP@16
C3: Qwen3-32B → Qwen3-1.7B ($N_M/N_S = 19.3$)						
A	20.4±0.0	9.5±0.1	11.3±0.6	12.3±0.8	11.0±0.4	12.3±0.4
B	24.9±0.1	9.7±0.2	12.0±0.7	12.8±0.7	11.4±1.0	14.0±1.3
C	25.1±0.1	10.2±0.1	12.5±0.4	13.7±0.6	11.0±0.5	14.3±0.5
D	69.7±0.2	56.6±0.1	59.2±0.2	58.5±0.2	57.9±1.0	60.4±0.7
E	34.0±0.0	20.7±0.1	22.7±0.7	22.1±0.7	22.7±0.8	23.6±0.8
F	47.3±0.1	28.9±0.0	30.0±0.4	29.7±0.3	30.3±0.3	30.0±0.5
G	89.8±0.0	73.1±0.1	74.1±0.2	70.5±0.8	74.3±0.3	79.3±0.2
H	78.3±1.1	64.5±1.3	66.1±1.0	62.1±0.2	65.5±1.0	68.0±0.6
I	17.6±0.1	7.7±0.1	9.5±0.6	10.7±0.9	8.8±0.3	10.7±0.8
J	76.0±0.1	46.3±0.1	51.4±0.3	54.6±0.4	49.1±0.3	55.1±0.7
K	82.7±0.7	68.3±0.7	71.4±0.7	66.1±1.5	69.4±0.8	72.1±0.3
L	81.3±0.8	67.0±0.8	69.2±0.7	64.8±0.3	67.9±0.3	70.5±0.7
M	65.4±1.5	51.6±0.9	53.0±0.7	49.2±0.8	53.4±1.2	56.7±2.0
Macro rec. (%)		—	14.5	5.9	9.6	26.8

Table 13: Full per-dataset results, pairs D1 and D2.

	M-ONLY	S-ONLY	T2T	C2C	LoRA	MP@16
D1: Qwen3-14B → Qwen3-8B ($N_M/N_S = 1.8$)						
A	20.2±0.1	6.1±0.1	11.1±0.3	13.9±0.5	8.7±0.4	13.5±0.4
B	20.6±0.1	13.5±0.2	15.6±1.0	16.6±1.1	14.5±1.8	18.4±0.7
C	18.6±0.1	12.0±0.0	14.1±0.9	15.3±0.3	13.3±0.8	15.4±0.5
D	68.4±0.0	63.5±0.0	64.9±0.6	66.0±0.3	64.7±0.8	66.6±0.8
E	32.8±0.2	28.6±0.1	29.6±1.0	29.4±1.2	28.7±0.6	30.8±1.2
F	42.7±0.1	35.9±0.1	35.8±1.6	35.8±0.8	36.6±0.3	36.5±1.0
G	88.3±0.0	82.8±0.1	83.6±0.2	81.1±0.4	83.8±0.6	87.0±0.6
H	75.9±0.8	70.3±0.9	71.2±0.6	68.4±0.7	71.0±0.7	73.6±1.1
I	16.4±0.2	12.1±0.1	14.1±0.3	15.5±0.5	12.9±0.5	15.3±0.9
J	73.8±0.1	67.1±0.1	69.7±0.3	71.1±0.7	68.3±0.4	71.3±0.2
K	82.9±0.9	76.7±0.9	78.5±0.7	74.3±0.9	77.7±0.6	80.5±1.3
L	78.0±0.5	73.5±0.6	75.4±1.4	71.9±0.7	74.0±0.6	76.2±0.8
M	62.8±1.5	57.9±0.9	59.7±0.7	55.1±0.5	58.8±0.7	60.4±0.7
Macro rec. (%)		—	28.6	12.3	15.6	57.1

	M-ONLY	S-ONLY	T2T	C2C	LoRA	MP@16
D2: Qwen3-14B → Qwen3-4B ($N_M/N_S = 3.7$)						
A	20.2±0.1	13.0±0.2	15.2±0.5	16.3±0.2	14.0±0.3	15.9±0.2
B	20.6±0.1	17.8±0.0	19.4±0.8	18.4±0.5	18.4±0.3	18.6±0.5
C	18.6±0.1	14.7±0.0	15.8±0.7	16.6±0.2	15.6±0.4	16.4±0.2
D	68.4±0.0	60.1±0.1	62.0±1.1	63.0±0.3	61.1±0.6	63.8±0.9
E	32.8±0.2	24.8±0.1	26.6±0.4	26.7±0.6	26.5±0.8	28.6±0.5
F	42.7±0.1	31.6±0.1	32.1±0.9	31.6±0.2	32.5±0.6	32.4±0.2
G	88.3±0.0	79.6±0.1	80.7±0.4	77.3±0.4	80.3±0.2	85.4±0.5
H	75.9±0.8	68.4±0.4	69.9±0.7	66.5±0.7	70.1±0.8	71.5±0.8
I	16.4±0.2	10.6±0.2	12.1±1.3	14.3±0.9	11.5±1.6	12.9±0.3
J	73.8±0.1	59.3±0.1	63.4±0.3	66.3±0.4	62.1±0.7	66.9±0.2
K	82.9±0.9	73.4±1.2	75.8±0.5	71.5±0.8	75.8±0.6	78.4±1.4
L	78.0±0.5	70.8±1.3	72.7±0.8	69.3±1.4	71.6±0.9	74.2±0.7
M	62.8±1.5	55.8±0.6	57.4±0.6	54.7±1.3	56.2±0.9	58.8±1.3
Macro rec. (%)		—	25.2	13.7	16.0	42.7

Table 14: Full per-dataset results, pairs D3 and E1.

	M-ONLY	S-ONLY	T2T	C2C	LoRA	MP@16
D3: Qwen3-14B → Qwen3-1.7B ($N_M/N_S = 8.7$)						
A	20.2±0.1	9.5±0.1	11.0±0.5	12.7±0.7	10.5±0.3	12.3±0.5
B	20.6±0.1	9.7±0.2	12.6±1.1	12.4±0.5	10.1±1.1	12.9±0.3
C	18.6±0.1	10.2±0.1	11.8±0.9	12.6±0.4	11.1±0.3	12.3±0.6
D	68.4±0.0	56.6±0.1	59.0±0.5	58.6±0.3	57.8±0.2	60.1±0.4
E	32.8±0.2	20.7±0.1	22.9±0.7	23.2±1.6	23.2±1.5	24.5±0.6
F	42.7±0.1	28.9±0.0	29.2±1.4	28.9±1.0	29.7±0.9	29.4±1.1
G	88.3±0.0	73.1±0.1	74.1±0.3	71.3±0.4	74.5±0.5	79.1±0.1
H	75.9±0.8	64.5±1.3	65.9±0.2	62.1±0.7	65.6±0.7	68.0±0.5
I	16.4±0.2	7.7±0.1	8.6±1.1	10.2±0.6	9.0±1.1	10.2±0.7
J	73.8±0.1	46.3±0.1	51.0±0.3	54.6±0.5	48.5±0.4	55.7±0.4
K	82.9±0.9	68.3±0.7	69.5±0.4	67.4±1.3	70.5±0.7	72.3±1.1
L	78.0±0.5	67.0±0.8	69.0±0.5	65.3±0.4	68.4±0.8	69.9±0.4
M	62.8±1.5	51.6±0.9	53.8±1.3	50.0±1.4	53.1±1.8	54.9±0.8
Macro rec. (%)		—	14.8	8.5	11.0	27.8

	M-ONLY	S-ONLY	T2T	C2C	LoRA	MP@16
E1: Ministral 3 14B → Ministral 3 8B ($N_M/N_S = 1.8$)						
A	42.7±0.0	33.1±0.1	35.9±0.4	36.7±0.7	35.0±0.5	37.9±0.3
B	38.9±0.1	31.9±0.1	33.1±0.4	35.1±0.3	31.4±0.9	35.6±1.7
C	43.4±0.0	37.7±0.1	38.9±0.3	40.0±0.2	38.3±0.4	40.7±0.4
D	15.5±0.1	8.8±0.1	10.5±0.8	11.2±0.6	10.1±0.7	11.8±0.9
E	4.3±0.1	0.9±0.1	2.5±0.6	1.7±0.7	1.4±0.3	3.1±1.5
F	47.1±0.0	40.9±0.1	41.7±0.7	41.4±1.0	41.3±0.6	41.2±1.1
G	71.9±0.2	65.9±0.2	66.7±0.1	64.7±0.2	66.9±0.5	70.5±0.6
H	79.3±0.6	72.6±0.4	74.3±0.3	70.4±1.2	74.4±1.0	75.6±0.3
I	10.4±0.2	6.4±0.1	7.1±1.2	8.5±1.0	6.5±1.0	9.6±0.3
J	76.8±0.1	69.9±0.1	72.5±0.7	73.6±0.4	70.8±0.2	73.9±0.2
K	75.1±0.8	69.6±1.1	71.3±0.7	66.7±0.8	70.4±0.4	72.9±0.3
L	79.4±1.1	73.2±0.8	74.9±0.9	71.3±0.4	74.1±0.4	76.3±0.4
M	60.8±1.5	55.6±1.1	57.6±0.6	54.9±0.3	57.1±0.5	58.2±1.2
Macro rec. (%)		—	26.4	11.3	13.9	53.0

Table 15: Inference-time refresh-interval sweep on A2 (mean±sd). * marks cells where the output ends before the first refresh boundary, so the condition is the static bridge exactly.

	R=1	R=2	R=4	R=8	R=16	R=32	R=64
A	36.5±0.2	36.5±0.3	36.7±0.7	36.7±0.8*	36.7±0.8*	36.7±0.8*	36.7±0.8*
B	34.8±0.5	35.2±0.4	34.7±1.5	34.9±0.4	34.7±0.4	33.9±1.4	34.0±1.5
C	51.0±0.8	51.1±0.7	50.9±0.6	51.0±0.4*	51.0±0.4*	51.0±0.4*	51.0±0.4*
D	60.2±0.6	60.5±0.3	60.3±0.5	60.0±0.5	60.2±0.3	59.7±0.3	57.9±0.8
E	24.2±1.1	23.8±0.4	23.8±0.5	23.9±1.1	23.7±1.1	22.7±1.2	21.5±0.6
F	37.4±0.9	37.1±0.8	36.7±0.4	37.2±0.6	37.0±1.2	37.1±0.9	36.6±0.4
G	90.1±0.3	89.5±0.4	89.4±0.4	89.3±0.3	88.9±0.4	87.5±0.1	85.1±0.3
H	75.9±1.1	76.3±0.5	76.0±1.5	75.6±1.0	75.3±1.0	74.2±0.8	71.6±0.8
I	17.8±1.1	17.6±1.5	17.3±0.5	17.6±0.9*	17.6±0.9*	17.6±0.9*	17.6±0.9*
J	81.4±1.1	81.3±0.4	81.2±0.4	81.3±0.6*	81.3±0.6*	81.3±0.6*	81.3±0.6*
K	83.4±1.0	83.6±0.5	83.4±0.4	83.6±0.8	82.9±1.3	81.0±0.5	79.5±1.0
L	80.8±1.5	80.8±0.4	80.7±1.1	80.7±1.3	79.8±0.8	78.4±0.6	76.0±0.4
M	65.4±1.6	65.5±0.5	65.4±0.9	65.0±0.9	64.4±1.0	62.4±1.2	60.0±0.8
Macro rec. (%)	57.1	56.8	55.3	55.0	52.2	43.1	29.4

Table 16: Macro-average recovery (%) against R for all eleven pairs. †The A2 row is derived from Table 15 and cross-checked against the registered sweep table by the derivation script.

Pair	R=1	R=2	R=4	R=8	R=16	R=32	R=64
A1	56.8	56.4	56.6	55.4	54.0	44.6	29.7
A2†	57.1	56.8	55.3	55.0	52.2	43.1	29.4
A3	37.9	37.5	37.5	36.8	35.7	29.2	23.8
B1	45.1	44.3	43.9	44.2	42.1	35.3	28.0
C1	57.5	56.9	57.6	56.6	54.6	43.5	31.3
C2	48.8	48.2	47.9	47.6	45.8	37.2	29.7
C3	28.6	28.5	28.0	26.6	26.8	22.7	16.3
D1	60.8	60.6	60.0	58.9	57.1	49.2	41.6
D2	45.6	45.3	45.0	44.0	42.7	37.8	31.2
D3	29.8	29.9	29.0	28.0	27.8	22.1	17.0
E1	55.9	56.8	55.1	53.7	53.0	46.2	36.1

F.4 rT2T Full Sweep

Table 17 is the complete rT2T quality sweep (A2; protocol in Appendix C.2). The nine triggering tasks (output $\geq R$) are swept over $R \in \{8, 16, 32\}$ and $L'_g \in \{64, 128\}$; the four short-answer sets trigger zero refreshes and equal T2T identically, so they carry no rows. The rT2T bar of Figure 4 takes the per-task best L'_g at R=16 (bold), giving the macro-recovery triplet T2T 26.9%, rT2T@16 33.7%, MP@16 52.2%. Three readings:

- **Best L'_g splits by task type.** G prefers 64 — on a strict-format task, longer injected text disturbs the format state more; the content tasks (B, D, E, H, K, L, M) prefer 128; F is flat against T2T (capability-limited, the "above T2T" expectation does not apply, consistent with the F exception throughout the paper). The split is stable only at R=16: at R=32 the 64 configuration of H edges past 128, inside noise.
- **Significance.** MP@16 over rT2T@16 (best) passes $2\sigma_{ab}$ only on B (+2.8, threshold 2.53) and G (+3.6, threshold 0.89); the other triggering tasks are within noise and are reported as comparable. rT2T over T2T is likewise significant only on B (+3.6) and G (+0.9): periodic refresh in text space points the right way but carries little, which is precisely the bandwidth argument for a latent refresh interface. Wherever rT2T's quality is described as comparable to MP's, the cost side must be cited in the same breath: at R=16 rT2T costs 41–54× MP@16 (Table 27).
- **Accumulation variant.** Accumulating hints instead of replacing them scores slightly lower everywhere and degrades as n_{sync} grows (context inflation ≈5.6k tokens on GovReport at R=16, L'_g=128; representative cell 81.0 ± 0.4, -0.6 vs. replacement); the full accumulation table will be released with the code and full logs.

F.5 Read-Distribution Diagnostic V_{64}

V_{64} is collected from the first 64 decoded steps with the bridge attached: 32 samples per dataset; the attention of each injection layer over the memory slots is averaged over heads to give the read distribution $p_{l,t}$; its variance is averaged over layers, steps, samples, and datasets. H_{64} is the Shannon entropy (nats) of the same distributions, the robustness twin of the variance version (high variance ↔ low entropy). Table 18 lists both for the eleven pairs next to their MP@16 macro recovery; the Spearman rank correlation is $\rho = 0.78$ ($p = 0.004$, $n = 11$). Computing V_{64} needs minutes of GPU time per pair, which is what makes it usable as a pre-deployment fit check; the three limits stated in Section 5.4 apply.

F.6 Short-Answer Controls

The two short-answer control columns (A, J) of the ablation are kept in the main text (Table 1); their standard deviations are in Table 9 and their non-triggering registration in Appendix A.2. This subsection is the designated landing spot for those columns should the main-text table need to shed them for space.

G Diagnostic Suite Details

The diagnosis of Section 3 runs on a dedicated suite, separate from the thirteen benchmarks; this section registers the suite and the full numbers behind Figure 2.

G.1 Suite Registry and Inclusion Criteria

Table 19 registers the five diagnostic tasks. Three criteria governed inclusion: (i) disjoint from the thirteen evaluation sets A–M, so the diagnostic and method lines share no data; (ii) disjoint from the training set fuse_v3; (iii) selected by prior expectation of staleness sensitivity — tasks whose targets keep shifting as generation proceeds. Criterion (iii) is the registered basis for the bridging statement in Section 5.3: the zero-training repair is larger on the diagnostic suite (+4.5 on D-IF) than on the benchmarks (+1.2 on IFEval), because the suite was chosen to be maximally staleness-sensitive, and the shallower expected representation decay on benchmark tasks (relative to the 0.32 probe depth on D-IF, Appendix G.4) corroborates it. D-MCQ (output median $9 < 16$) is the suite's only strictly zero-trigger task at R=16 and carries every "non-triggering" statement of Section 3; the benchmark-side short-answer facts are registered separately in Appendix A.2 and the two registrations do not overlap. Tokenizer and length conventions, seeds, and decoding match Appendix A.3.

Table 17: rT2T full quality sweep on A2 (replace-style; mean±sd). Bold marks the per-task best L'_g at R=16, the configuration entering Figure 4. Short-answer sets (A, C, I, J) trigger no refresh and equal T2T cell by cell, as does the $R\to\infty$ limit (double identity, hard-verified).

Task	L'_g	R=8	R=16	R=32	T2T ($= R\to\infty$)	MP@16
B GPQA	64	31.4±0.9	31.6±0.6	31.2±0.7	28.3±0.8	34.7±0.4
	128	32.0±0.5	**31.9±1.2**	31.5±0.5	28.3±0.8	34.7±0.4
D MATH-500	64	59.9±0.9	59.7±0.6	59.9±0.2	59.4±0.4	60.2±0.3
	128	60.2±0.6	**59.9±0.4**	59.6±0.2	59.4±0.4	60.2±0.3
E OlympiadBench	64	23.1±1.0	22.7±0.6	22.4±0.3	21.9±0.6	23.7±1.1
	128	22.8±1.1	**23.0±0.4**	22.5±1.0	21.9±0.6	23.7±1.1
F LiveCodeBench	64	38.3±0.4	38.0±0.4	38.1±0.6	38.0±1.2	37.0±1.2
	128	37.9±0.8	**38.3±0.6**	37.9±0.6	38.0±1.2	37.0±1.2
G IFEval	64	85.7±0.6	**85.3±0.2**	85.1±0.2	84.4±0.2	88.9±0.4
	128	84.8±0.5	84.9±0.6	84.4±0.3	84.4±0.2	88.9±0.4
H WritingBench	64	73.3±0.6	73.0±0.3	73.3±0.9	72.5±0.5	75.3±1.0
	128	73.5±0.2	**73.3±0.8**	73.1±0.6	72.5±0.5	75.3±1.0
K GovReport	64	81.2±0.7	81.4±1.0	80.9±0.4	80.5±0.4	82.9±1.3
	128	81.6±1.0	**81.6±1.2**	81.3±0.6	80.5±0.4	82.9±1.3
L MultiNews	64	78.3±0.6	78.2±0.7	77.8±0.3	77.9±0.4	79.8±0.8
	128	78.5±0.8	**78.6±1.6**	78.4±0.5	77.9±0.4	79.8±0.8
M LongBench-Write	64	63.1±0.4	63.1±1.0	62.9±0.6	62.4±0.8	64.4±1.0
	128	63.0±1.6	**63.2±1.6**	62.8±1.4	62.4±0.8	64.4±1.0

Table 18: Read-distribution diagnostics for the eleven pairs: variance version V_{64}, entropy version H_{64}, and MP@16 macro recovery. Spearman $\rho(V_{64}, \text{rec.}) = 0.78$, $p = 0.004$.

Pair	V_{64} ($\times 10^{-3}$)	H_{64} (nats)	MP macro rec. (%)
A1	3.03±0.19	4.40±0.07	54.0
A2	3.26±0.05	4.26±0.04	52.2
A3	2.38±0.36	4.54±0.06	35.7
B1	2.73±0.20	4.44±0.10	42.1
C1	3.10±0.06	4.23±0.03	54.6
C2	2.42±0.08	4.58±0.11	45.8
C3	1.97±0.23	4.66±0.14	26.8
D1	3.28±0.10	4.28±0.03	57.1
D2	2.82±0.24	4.45±0.04	42.7
D3	3.00±0.19	4.28±0.22	27.8
E1	3.42±0.23	4.08±0.08	53.0

G.2 Main Condition Matrix

Table 20 is the full condition matrix with standard deviations. The sign flip is asserted only on D-IF, where OneShot − S-only $= -2.5$ exceeds twice the pooled standard deviation (0.60); D-SUM (-0.5) and D-WR (-1.2) move in the same direction below threshold and are reported as same-direction only. OracleSwap@16 repairs $+4.5$/$+2.7$/$+3.1$ over OneShot on the three staleness-sensitive tasks and clears the floor by $+2.0$ on D-IF. The OracleSwap-mis@16 row — initial memory matched, refresh points swapped with fresh memory of *another* sample — shows no repair anywhere (≈ OneShot or slightly below), pinning the repair on content freshness rather than the swap action. On D-CODE every condition sits within noise: a capability failure refresh cannot fix. On D-MCQ the three initially-matched @16 conditions equal OneShot at 87.9 exactly (zero-trigger identity). OneShot-mismatch is collected only on D-MCQ, where it serves as the content canary: the 5.4-point drop (82.5 vs. 87.9) shows the student reads the memory content even when no refresh ever fires; on the triggering tasks content attribution is already carried by the OracleSwap-mis@16 row, so no duplicate collection is made. The drop is direction-consistent with ablation row 5 on MMLU-Pro (-12.9 vs. the full system); the magnitude gap follows from D-MCQ's near-saturated anchors.

G.3 Position-Segment Quality

Table 21 gives the per-segment constraint satisfaction on D-IF behind Figure 2a. OneShot starts slightly above the floor, crosses below it in the middle third, and collapses in the final third, while OracleSwap@16 holds the late segment; the crossing is the "turns harmful" point of Section 3. Each condition's three segment means reproduce its aggregate in Table 20, a consistency check enforced at fill time.

G.4 StaleGap Probe Protocol

The probes are lightweight linear readers trained on frozen mentor states to predict verifiable properties of the next fixed-length window (for D-IF, which constraints the window will satisfy). Fresh and stale probes share architecture and training budget and differ only in their inputs, $H_{\mathcal{M}}(x \oplus \hat{y}_{<t})$ versus $H_{\mathcal{M}}(x)$; they act on mentor states directly and do not involve the bridge, so the measured decay is a property of the guidance signal, not of the transfer mechanism. Table 22

Table 19: The diagnostic suite (D suite): disjoint from the benchmarks A–M and from fuse_v3, selected by prior staleness sensitivity.

Code	Dataset (role)	Metric	Size	In. med.	Out. med.
D-IF	Multi-IF (primary: sign flip, PosQ)	constraint satisfaction	400	120	610
D-SUM	QMSum (long-in/long-out positive)	LLM-judge mean $\times 10$	200	8200	640
D-WR	HelloBench (long writing) (open-ended positive)	LLM-judge mean $\times 10$	150	190	1150
D-CODE	BigCodeBench (capability-limited negative)	pass@1	300	380	450
D-MCQ	ARC-Challenge (non-triggering negative)	accuracy	1170	460	9

Table 20: Diagnostic condition matrix on A2 (mean±sd). OneShot-mismatch is collected only on D-MCQ (content canary); on triggering tasks attribution is carried by OracleSwap-mis@16.

Condition	D-IF	D-SUM	D-WR	D-CODE	D-MCQ
M-only (ceiling)	73.2±0.2	68.9±0.9	71.3±0.5	47.6±0.1	92.8±0.1
S-only (floor)	58.7±0.0	59.4±0.7	62.7±0.6	30.8±0.1	85.8±0.1
OneShot (static bridge)	56.2±0.3	58.9±0.4	61.5±0.5	30.5±0.2	87.9±0.6
OracleSwap@16	60.7±0.5	61.6±0.3	64.6±0.2	31.1±0.5	87.9±0.6
OracleSwap-mis@16	55.5±0.8	58.7±0.2	61.0±0.5	30.2±0.9	87.9±0.6
OneShot-mismatch	—	—	—	—	82.5±0.4

Table 21: Position-segment quality (PosQ) on D-IF; the aggregate column equals Table 20.

Condition	First 1/3	Middle 1/3	Last 1/3	Aggregate
S-only	59.0±0.3	58.6±0.3	58.3±0.4	58.7
OneShot	59.8±0.3	56.4±0.3	52.6±0.8	56.2
OracleSwap@16	59.7±0.8	60.5±1.1	61.8±0.6	60.7

registers the curves: the stale readability decays from 0.79 to 0.47 over 512 generated tokens (decay depth 0.32) while the fresh readability stays within 0.76–0.80; Figure 10 in Appendix I.2 plots them with the resulting $\mathrm{StaleGap}(t)$.

Table 22: StaleGap probe readability on D-IF (linear readout, independent of the bridge). $\mathrm{StaleGap}(t)$ is the fresh−stale difference.

t	0	64	128	256	384	512
stale $H_{\mathcal{M}}(x)$	0.79	0.73	0.69	0.58	0.53	0.47
fresh $H_{\mathcal{M}}(x \oplus \hat{y}_{<t})$	0.79	0.80	0.78	0.77	0.76	0.78
$\mathrm{StaleGap}(t)$	0.00	0.07	0.09	0.19	0.23	0.31

G.5 Identity and Decoupling Constraints

Four constraints are enforced mechanically on every diagnostic and sweep table. (i) *Zero-trigger identity:* whenever a task's output median is below R, every initially-matched @R condition (OracleSwap@R, OracleSwap-mis@R, MP@R, rT2T@R) must equal its static counterpart cell by cell; the fill scripts hard-verify this and refuse to write violations. Mismatch-induced drops may therefore appear only under static-mismatch conditions. (ii) *Line decoupling:* the diagnostic tables share no cell with the benchmark tables (S1/S3/S5 lines); the diagnosis and the method evaluation are numerically independent. (iii) *Significance discipline:* definite-language claims appear only where a difference exceeds twice the pooled standard deviation (Appendix A.4). (iv) *Anchor consistency:* the suite's floor and ceiling levels sit at the ability levels the A2 pair shows on comparable benchmarks under the full protocol, cross-checked against the capability ordering of the screening pass (Appendix A.3).

H Cost Accounting

H.1 Byte Accounting

Each transmitted slot is one d_t-dimensional bf16 vector. For refresh interval R, slot granularity S=32, tail size W_{tail}=32, and k transmitted layers (top-k by ρ_o), the amortized payload per update is

$$\mathrm{Bytes}(R) = k\,(R/S + W_{\mathrm{tail}})\,d_t \times 2 \text{ bytes} \qquad (3)$$

(a new summary slot completes only every $\lceil S/R \rceil$-th refresh; the worst single update ships $\lceil R/S \rceil$ summary slots), dominated by the tail term, so the per-refresh payload is nearly constant over $R \in [1, 64]$ and the left-end cost divergence of Figure 8 comes from synchronization *count*, not payload growth. Older summary slots are retained on the student side and never retransmitted. The initial memory costs $k(p_x + W_{\mathrm{tail}})d_t \times 2$ bytes with $p_x \le P$=128, capping the initial transfer regardless of input length. Table 23 lists the accounting for the A2 configuration (d_t=2560) and the C2C comparison point.

H.2 Per-Request Cost Formulas

Let $P(\cdot)$ and $D(\cdot)$ be measured prefill and decode throughputs, $D_b(\mathcal{S})$ the student's decode throughput with the bridge attached, and c_M, c_S the rental rates of the mentor's and student's GPUs. With $n_{\mathrm{sync}} = \lfloor (L_{\mathrm{out}} - 1)/R \rfloor$ (zero when $L_{\mathrm{out}} < R$: short outputs never trigger) and $\tau_{\mathrm{sync}}(R) =$

Table 23: Transfer accounting at R=16 for A2 (d_t=2560; 150 Mbps cross-site link, 18 ms RTT). The k=16 row is the deployed configuration (ablation row 11).

Layers k	Bytes(16)	Transfer	τ_{sync} (w/ RTT)
8	1.33 MB	71 ms	89 ms
16 (main)	2.66 MB	142 ms	160 ms
32	5.32 MB	284 ms	302 ms
Initial memory (p_x=128, k=16)			13.1 MB
Full fused KV (C2C, 2K input)			∼268 MB

$$\text{Bytes}(R)/\text{bandwidth} + \text{RTT},$$

$$\text{Cost}_{MP}(R) = c_M \frac{L_{\text{in}}+L_{\text{out}}}{P(\mathcal{M})} + c_S\Big[\frac{L_{\text{in}}}{P(\mathcal{S})} + \frac{L_{\text{out}}}{D_b(\mathcal{S})} + n_{\text{sync}}\,\tau_{\text{sync}}(R)\Big], \tag{4}$$

$$\text{Cost}_{\text{T2T}} = c_M\Big[\frac{L_{\text{in}}}{P(\mathcal{M})} + \frac{L_g}{D(\mathcal{M})}\Big] + c_S\Big[\frac{L_{\text{in}}+L_g}{P(\mathcal{S})} + \frac{L_{\text{out}}}{D(\mathcal{S})}\Big]. \tag{5}$$

Two structural facts follow. Incremental prefill touches each generated token exactly once, batched into one parallel forward per refresh, so the mentor's total compute is independent of R; only the synchronization term grows as R shrinks. And T2T pays $L_g/D(\mathcal{M})$, mentor decoding, the most expensive per-token operation in the pipeline, which MentorPulse never performs. rT2T (replace-style) turns that one-time decoding tax into a per-refresh tax:

$$\text{Cost}_{\text{rT2T}}(R) = \text{Cost}_{\text{T2T}} + c_M\Big[\frac{L_{\text{out}}}{P(\mathcal{M})} + n_{\text{sync}}\frac{L'_g}{D(\mathcal{M})}\Big] + c_S\Big[\sum_{j=1}^{n_{\text{sync}}} \frac{L'_g + L_{\text{in}} + jR}{P(\mathcal{S})} + n_{\text{sync}}\Big(\frac{L'_g}{D(\mathcal{M})} + \text{RTT}\Big)\Big], \tag{6}$$

where the sum is the replace-style full re-prefill and the second c_S term is the student idling while the mentor decodes each hint; the accumulation variant replaces the sum by $n_{\text{sync}}L'_g/P(\mathcal{S})$. The mentor decoding term dominates (>80% of total), which is why the replace-vs-accumulate cost difference is second order (Appendix C.2).

H.3 Measured Throughput and Prices

All throughputs are measured with vLLM 0.19.1 in bf16 at batch size 1 and 8K-token context on the deployment stack of Section 5.1; rental rates are public on-demand quotes with their retrieval date. Costs are reported after converting to \$/Mtok (dollars per million output tokens) using each task's measured L_{out}; the conversion never changes a within-task dominance verdict.

Table 24: Measured serving throughput (vLLM 0.19.1, bf16, batch size 1, 8K context). D_b is decoding with the bridge attached.

Model	Hardware	P	D	D_b
Qwen3.5-27B (mentor)	H100 80GB	8,600	54	—
Qwen3.5-4B (student)	RTX 4090 24GB	23,500	121	104

Table 25: GPU rental prices used in the cost model.

Hardware	\$/h	Source	Quoted
H100 80GB (mentor)	2.79	Lambda on-demand	2026-07-10
RTX 4090 (student)	0.44	RunPod community	2026-07-10

H.4 Synchronization Parameters

The main accounting uses a cross-site leased line, matching the deployment narrative of a data-center mentor and an edge student: 150 Mbps effective bandwidth, 18 ms round-trip latency, blocking synchronization, and k=16 transmitted layers (consistent with ablation row 11). One measured refresh at R=16 takes 165 ms end to end. The measurement is the student-observed blocking window and covers every synchronization step: shipping the R new tokens to the mentor, the mentor's incremental prefill of those tokens (one batched parallel forward, not decoding), slot construction, and the slot transfer with its round trip. Because a layer's slots are serialized out as soon as that layer's forward and projection complete, the forward and construction overlap the 142 ms transfer, and the transfer-dominated formula estimate of 160 ms (142 ms transfer + 18 ms RTT) lands within 3% of the measured window; the priced tables use the measured 165 ms as τ_{sync}. Since $\text{Bytes}(R)$ is tail-dominated, the per-refresh time is nearly constant across $R \in [1, 64]$. This is also why Eq. 6 carries an explicit idle term while Eq. 4 does not: at a refresh MP's mentor runs one batched forward over R tokens inside the 165 ms window, whereas rT2T's mentor must decode $L'_g \in \{64, 128\}$ tokens sequentially — 64–128 dependent forwards — blocking the student for 1.20–2.39 s (Appendix H.5); the asymmetry is in the mentor-side operation, not in the accounting. The mentor is billed by *active compute seconds* (a shared serving pool; the mentor is not resident per request); the resident-billing alternative is in Appendix H.6.

H.5 Cost Support Numbers

Table 26 gives the per-request costs behind the dominance verdicts of Figure 8 (blocking accounting; quality inputs from Tables 8 and 15, throughputs and prices from Tables 24 and 25, lengths from Table 4). The left-end divergence on the long-output rows comes entirely from the synchronization term ($n_{\text{sync}} \propto 1/R$). Dominance is judged on strict numeric comparison per Section 5.6; rT2T enters no dominance panel because it is cost-dominated at every R (each refresh pays a mentor decoding tax), so Figure 8 keeps its two-method verdict clean and rT2T is cited from Table 27 instead. At R=16 with the quality-best L'_g=128, rT2T costs 17–28× T2T and 41–54× MP@16 on the two long-output representatives; a single refresh blocks the student for 1.20 s (L'_g=64) or 2.39 s (128) of mentor decoding versus MP's 0.165 s, and a GovReport request accumulates 52.9–105.1 s of blocking versus 7.3 s for MP@16. The structural contrast is worth stating: rT2T's left-end divergence comes from the per-refresh decoding tax $n_{\text{sync}}L'_g/D(\mathcal{M})$, MP's from the transfer term — text refresh is expensive because the mentor must *speak* at every refresh, MP only because it ships bytes more often.

Table 26: Per-request cost (10^{-3} dollars, blocking accounting) for the four representative tasks of Figure 8.

Task	T2T	MP@1	MP@2	MP@4	MP@8	MP@16	MP@32	MP@64	MP@∞
A	1.91	0.15	0.09	0.07	0.05	0.05	0.05	0.05	0.05
J	3.44	0.59	0.53	0.51	0.49	0.49	0.49	0.49	0.49
H	4.55	20.14	10.66	5.92	3.56	2.37	1.79	1.49	1.20
K	5.89	15.81	8.72	5.17	3.39	2.50	2.06	1.84	1.62

H.6 Sensitivity Analyses

Three accounting variations. *Network:* on a same-site gigabit LAN the synchronization term becomes negligible and the purple segments of Figure 8 extend left to $R \approx 2$ (analytic extrapolation of the cost side). *Synchronization mode:* asynchronous prefetch overlaps transfer with decoding and lowers the synchronization term further; the main figure conservatively uses blocking. *Mentor billing:* billing a resident, dedicated mentor for the full request duration instead of active compute seconds reverses the direction of the cost conclusion — a resident mentor idles between refreshes, and that idle time dwarfs the compute it sells; we report this reversal as is. The cost structures of C2C and LoRA are different in kind rather than in degree and are therefore discussed, not priced, in Figure 8: C2C ships a full fused KV cache (∼268 MB at a 2K input, growing with length) and presumes same-machine or same-rack co-location, while LoRA involves no online mentor at all — its serving cost is a strict lower bound among the compared methods, and Table 2 shows what that saving costs in quality.

I Additional Results and Case Studies

I.1 Per-Dataset Generality

Figure 9 expands Table 2 to the per-dataset level: eleven pairs × thirteen datasets for each of the four methods, on the recovery color scale of Figure 5. Three patterns repeat across every mentor group. The C2C panel shows the sign-flip columns (G, H, K, L, M) in red for all eleven pairs — the harm of static latent guidance on long outputs is not a property of one pair. The MP panel keeps those columns purple everywhere, with saturation fading within each mentor group as the capacity gap grows (the fit-interval pattern of Section 5.4). And the F column stays near white in all four panels: no interface rescues a capability the student lacks.

I.2 Probe Curves

Figure 10 plots the registered probe values of Table 22 (protocol in Appendix G.4): the stale curve decays from 0.79 to 0.47 (depth 0.32) while the fresh curve holds 0.76–0.80, and the widening wedge between them is $\mathrm{StaleGap}(t)$ — the guidance drifts away from the task it is supposed to describe even though the input never changes.

I.3 Learned Staleness Trigger (Stage-2 Outlook)

All experiments in this paper use a fixed refresh interval; a learned trigger (MP-ST) that decides *when* to refresh is future work, and we register here a preliminary, prospective observation from a stage-2 pilot. The trigger is trained on an offline refresh-value target: for every window, the cross-entropy difference between decoding under the stale and the fresh memory version in the versioned store. A lightweight predictor reads bridge statistics, including attention on recent tail slots, and its threshold is calibrated to match the refresh count of the fixed schedule, so the comparison is at equal budget. Figure 11 shows the qualitative behavior: firing concentrates at constraint switches and topic boundaries rather than uniformly, suggesting that a content-aware schedule could buy back part of the small-R quality at fixed cost. No collection table registers MP-ST numbers yet, and the figure makes no quantitative claim.

I.4 Qualitative Cases

Figure 12 shows two representative generation pairs, one from each side of the applicability boundary. In the success case (long instruction following), the student under OneShot begins correctly, then follows the stale memory back to an already satisfied constraint and never produces a later one; under MP@16 each refresh sees which constraints are done and the guidance tracks the ones still open. In the failure case

Table 27: rT2T per-request cost (10^{-3} dollars, replace-style, blocking; analytically computed from Eq. 6 and the measured inputs). Bold marks each task's quality-best configuration (Table 17). At ∞, n_{sync}=0 and rT2T equals T2T identically, as on the short-answer tasks (which therefore carry no rows).

Task	L'_g	R=4	R=8	R=16	R=32	R=64	∞	T2T	MP@16
H WritingBench	64	254.8	129.7	66.7	35.6	19.6	=T2T	4.55	2.37
	128	503.7	254.2	**128.3**	66.5	34.5	=T2T	4.55	2.37
K GovReport	64	200.8	103.4	54.7	30.3	18.1	=T2T	5.89	2.50
	128	388.0	197.0	**101.5**	53.7	29.8	=T2T	5.89	2.50

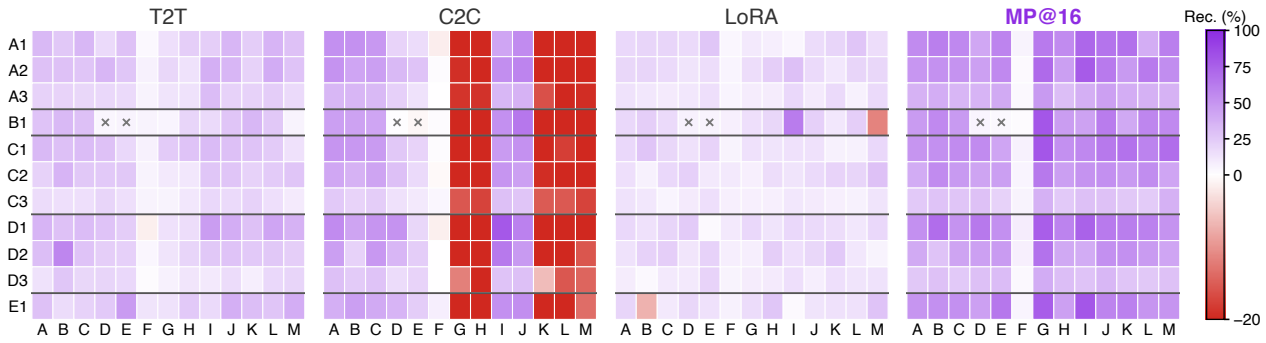

Figure 9: Per-dataset recovery for all eleven pairs (rows, grouped by mentor group) on all thirteen datasets (columns) under the four methods; color is recovery clipped to $[-20, 100]$ as in Figure 5, red marking harm. $\times$ marks the two capability-limited B1 cells registered in Appendix F.2. Numbers behind every cell are in Tables 8 and 10–14.

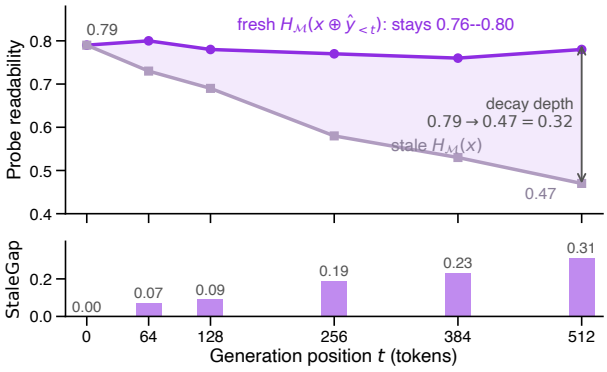

Figure 10: StaleGap probe on D-IF (enlarged version of Figure 2b; linear readout on frozen mentor states, independent of the bridge). Top: fresh and stale probe readability over generation position; the shaded wedge is $\mathrm{StaleGap}(t)$ and the bracket marks the 0.32 decay depth. Bottom: $\mathrm{StaleGap}(t)$ values (fresh − stale, Table 22).

Figure 11: Stage-2 outlook (illustrative pilot; no quantitative claim). Background: density of offline refresh-value labels (CE difference between decoding under stale and fresh memory). Bars: where the learned trigger fires at the R=16 equal budget; the dashed line is the uniform schedule. Firing concentrates at constraint switches; the inset shows the event-aligned rate on the R=16 window scale.

(code generation, the capability-limited mode), OneShot and MP@16 fail identically: every fresh hint restates a plan the student cannot execute better, so refresh repairs nothing — refresh fixes staleness, not missing capability, matching the F and D-CODE boundaries in the main text.

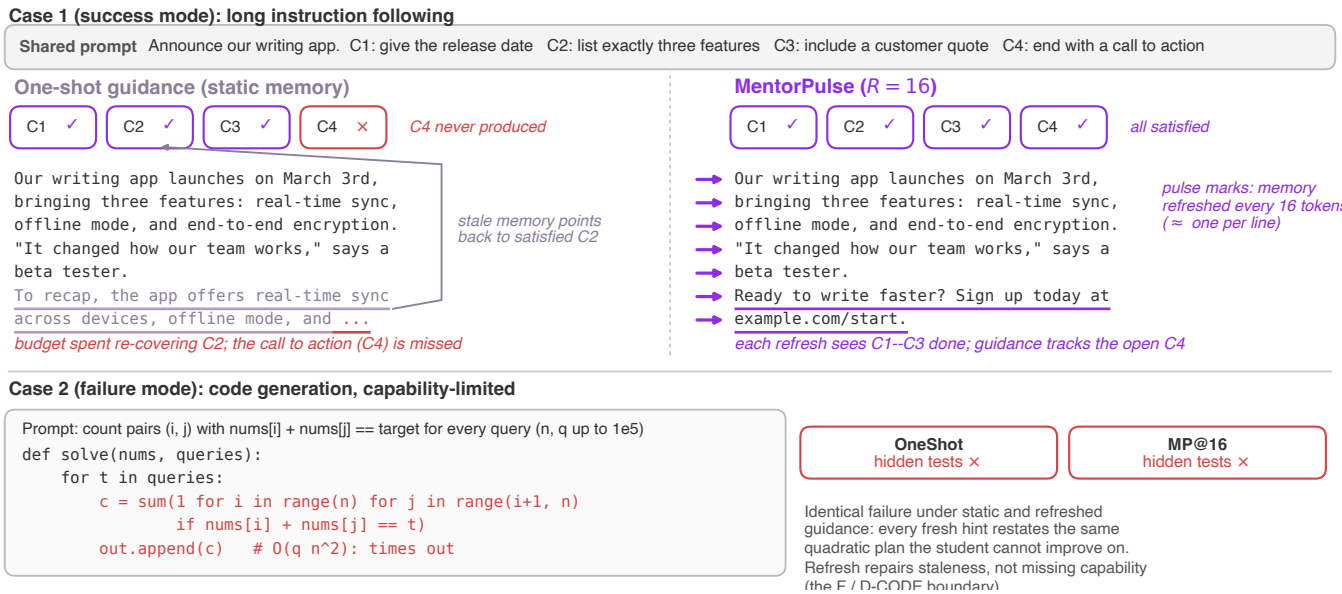

Figure 12: Two illustrative cases (more in the full logs, to be released). Top: on long instruction following, static guidance points the student back to a satisfied constraint (gray underline and arrow) and a later constraint is missed (red); with R=16 the refreshed memory reflects the text written so far and the remaining constraint is completed (purple). Bottom: on a capability-limited coding task both conditions produce the same flawed algorithm — fresh guidance does not create missing capability.