COMPRESS WHAT YOU SEE, NOT WHAT YOU SAY: ANCHORED CONTEXT DISTILLATION FOR LATENT-OBSERVATION SOFTWARE ENGINEERING AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool observations dominate the context of software-engineering agents, making long interaction histories costly to maintain. Existing context compression methods can discard information needed by later actions, while adapting agents to soft-token representations can compromise their original behavior. To reduce context while preserving action-critical information and agent behavior, we combine *Latent Observations, Hard Actions* (LOHA), a context layout that separates compressed history from text needed for exact reference, with *Anchored Context Distillation* (ACD), a training method that enables latent reading while constraining behavioral drift. LOHA compresses older tool observations into soft tokens while retaining the agent's own turns and the last K observations in text, providing compact access to historical information and exact access to recent content. To enable the agent to use this representation, ACD distills the base model's full-text predictions into the latent view while anchoring its behavior on plain-text inputs to the same base model. On SWE-bench Verified, $K{=}3$ reduces context per call by 43% for Qwen3-4B and 57% for SWE-Master-4B-RL, with resolve rates of 12.1% and 21.8% versus 14.5% and 27.5% for their uncompressed bases. A single-run recency sweep reaches 14.4% and 23.0% at $K{=}8$, with larger windows generally favoring task performance over compression. Under a 32K-token limit, Qwen3 with $K{=}3$ resolves 21.1% of a 199-instance subset versus 11.1% for the same adapted agent using full text. In concurrent single-GPU serving, it achieves $1.9\times$ that full-text agent's instance throughput.

1 INTRODUCTION

Large language models are increasingly used as software-engineering agents. To address a GitHub issue, an agent uses tools to inspect the repository, modify source code, and test its changes, refining the patch based on the results (Jimenez et al., 2024; Yang et al., 2024a; Xia et al., 2025; Wang et al., 2025). These interactions produce tool observations, such as file contents, search results, stack traces, and test logs, that accumulate in the agent's context. In CoderForge trajectories (Ariyak et al., 2026), tool observations account for 69% of trajectory tokens on average.

Existing approaches reduce context through observation masking (Lindenbauer et al., 2025), history summarization (Packer et al., 2023; Kang et al., 2026), sub-task folding (Sun et al., 2025; Ye et al., 2025), or selective pruning of tool outputs (Wang et al., 2026a;b; Ren et al., 2026; Chen et al., 2026). These methods shorten interaction histories, but deciding what to discard is difficult when later actions may depend on details in earlier observations. Coding agents also require more than a semantic summary: a `str_replace` edit, for example, requires an exact match to the source text. Information omitted during compression may therefore have to be retrieved again before the agent can act. Source-code compressors (Shi et al., 2025; Wang et al., 2024) exploit program structure to guide compression, but are less suited to the search results, stack traces, and test logs that also occupy a substantial portion of an agent's context.

Soft-token compression (Mu et al., 2023; Ge et al., 2024; Li et al., 2025; 2026) offers an alternative by encoding text into a shorter sequence of continuous embeddings that the decoder reads directly. For coding agents, however, a useful compressed representation must support both understanding

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

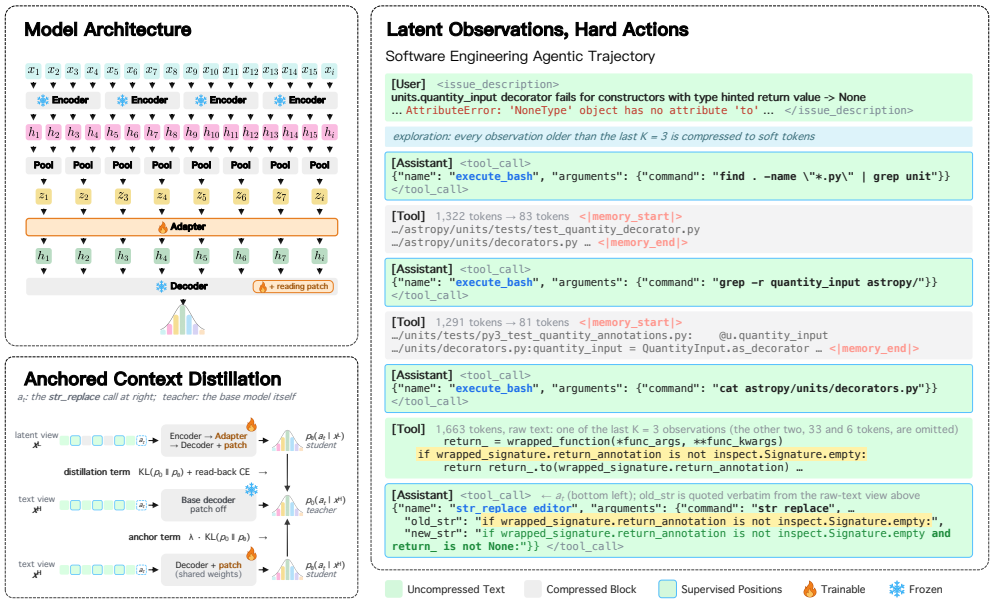


Figure 1: LOHA retains recent text and compresses older observations; ACD adds latent reading with a full-text behavioral anchor. Only the adapter, reading patch, and normalization gains are trained.

past observations and reproducing specific details for tool calls. These requirements are distinct: retaining enough information to reason about a file does not necessarily preserve the exact strings needed to edit it. Adopting soft-token representations also requires the decoder to learn a new input representation while retaining its existing ability to reason, invoke tools, and modify code. This raises two closely related questions: how should an agent combine compressed history with access to exact text, and how can it learn to use that history without compromising its ability to act?

We address these questions with a framework that combines context compression with behavior-preserving adaptation (Figure 1). Its two components, *Latent Observations, Hard Actions* (LOHA) and *Anchored Context Distillation* (ACD), determine how the agent's history is represented and how the agent learns to use it. LOHA follows a simple principle: **compress what the agent sees, not what it says**. It represents older tool observations as soft tokens while keeping the agent's own turns, the system prompt, and the task description in ordinary text. To support actions that require exact strings, it also retains the K most recent observations in raw text. This layout provides a compact representation of historical observations while preserving direct access to recent content for precise tool calls. ACD then enables the agent to read the compressed observations while limiting changes to its existing behavior. It combines two complementary objectives: a distillation term aligns predictions from the latent view with those of the frozen base model given the full text, while an anchor term constrains the adapted model's predictions on full-text inputs to remain close to the same base model. Together, LOHA and ACD address the representation and adaptation challenges of using compressed observations in coding agents.

We evaluate LOHA + ACD on SWE-bench Verified with Qwen3-4B-Instruct and its RL-fine-tuned derivative, SWE-Master-4B-RL. At their full context windows, $K{=}3$ reduces context per call by **43%** and **57%**, with resolve rates of **12.1%** and **21.8%** versus **14.5%** and **27.5%** for their uncompressed bases. In the single-run recency sweep, $K{=}8$ resolves 14.4% and 23.0%, versus 13.6% and 22.2% at $K{=}3$, with more text retained and higher compute cost (Table 3a). On SWE-Master, $K{=}3$ also outperforms masking (17.6%). Under a 32K-token limit, Qwen3 with $K{=}3$ resolves **21.1%** of a 199-instance subset versus **11.1%** for the adapted full-text agent. Concurrent single-GPU serving achieves 1.9× that agent's instance throughput. Our contributions are:

- **A context layout for latent-observation coding agents.** LOHA combines soft-token representations of older tool observations with uncompressed agent turns and a recent-observation window, providing compact historical context alongside exact text for actions involving recent content.

- **Anchored Context Distillation for behavior-preserving adaptation.** ACD combines cross-view distillation with a self-anchoring objective to teach latent reading while limiting behavioral drift. Our experiments show that better read-back performance alone does not ensure better agent behavior and identify self-anchoring as an effective way to balance the two.
- **An empirical evaluation of the performance–efficiency trade-off.** We compare task success and context usage on both an instruction-tuned agent and an RL-fine-tuned agent, and examine context limits and serving throughput. The results characterize the performance cost of compression and the settings in which it provides practical benefits.

2 APPROACH

Our framework combines two components: *Latent Observations, Hard Actions* (LOHA), which organizes the agent's context into compressed observations and uncompressed text, and *Anchored Context Distillation* (ACD), which trains the agent to use this representation while limiting changes to its existing behavior. We first define the context layout, then describe the student and its objectives, and finally explain training, inference, and cost accounting.

2.1 PROBLEM SETUP

An issue-resolution trajectory consists of a system prompt s containing tool schemas, an issue description u, assistant turns a_t, and tool observations o_t: $\tau = (s, u, a_1, o_1, \ldots, a_T)$. Assistant turns contain reasoning and tool calls; observations contain information returned by the environment, such as file contents, search results, and test logs.

We use an LCLM-based compression pipeline (Li et al., 2026), denoted by $\mathcal{E}$, comprising an encoder and an adapter that maps its output to the decoder's input space. At the default $16\times$ ratio, n text tokens become approximately $n/16$ continuous embeddings (*soft tokens*). The decoder processes these alongside ordinary text-token embeddings (*hard tokens*).

2.2 LATENT OBSERVATIONS, HARD ACTIONS

LOHA compresses older tool observations while retaining the system prompt, issue description, and all assistant turns in text. A recent-observation window also remains uncompressed, providing exact text for actions involving recently accessed content.

Rendering. Each observation remains in its original position within the decoder's native tool-response envelope. For an observation selected for compression, its body is wrapped in memory markers, encoded in 1,024-token windows, and replaced by soft tokens. The surrounding envelope remains in hard tokens, preserving its association with the tool call; no additional header is introduced. Unlike pruning, this represents the observation through an encoder rather than selecting text spans, though compression does not guarantee verbatim reconstruction.

The recency policy (Hard-Last-K). An edit often follows a file view, and a test command may quote an identifier from a recent failure report. LOHA therefore uses recency to retain raw text, rather than predicting which observation the next action will quote. Given m observations, the body of o_i is represented as

$$r_K(o_i \mid m) = \begin{cases} o_i, & i > m - K, \\ \mathcal{E}(o_i), & \text{otherwise.} \end{cases}$$

For $K > 0$, each new observation enters as text and becomes eligible for compression when it leaves the window. Older observations remain available through their latent representations; exact content can be retrieved again using the existing tools. The window size is selected at inference without retraining: increasing K retains more exact text but reduces compression for a fixed history. We use $K = 3$ for the main comparisons and evaluate $K \in \{0, 1, 2, 3, 4, 8, \infty\}$, spanning fully latent to full-text observations (Section 3.3).

2.3 ANCHORED CONTEXT DISTILLATION

ACD adapts a base agent to read latent observations while regularizing its predictions on ordinary text. We apply the same recipe to Qwen3-4B-Instruct-2507 (Qwen Team, 2025) and SWE-Master-4B-RL,

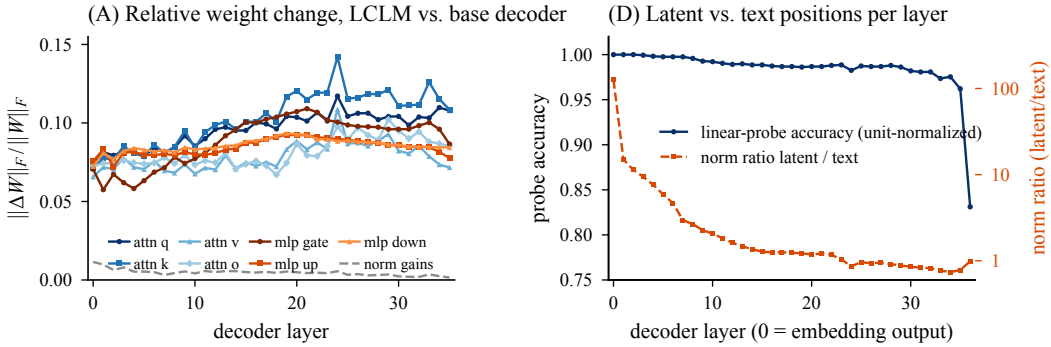


Figure 2: (A) Projection-weight and normalization-gain changes from Qwen3 to LCLM. (D) Unit-normalized latent/text probe accuracy and mean norm ratio. Full measurements: Appendix E.

each serving as the base decoder for its own student and as its frozen teacher. Rather than adopting the continually pre-trained LCLM decoder directly, we use its weight difference from the base decoder to initialize a low-rank reading patch.

The training objective covers two input views. On the latent view, the student learns to match the teacher's predictions from the corresponding full-text context. On the full-text view, the student is regularized toward the teacher's original predictions. These objectives address latent reading and behavioral preservation within the same adapted model.

Design motivation. Soft tokens introduce a new input representation, so the decoder must learn to use them while retaining its existing tool-use policy. The LCLM decoder supplies a trained reading initialization, but adopting it directly would also replace the base agent's decoder. A low-rank patch provides a limited set of parameters through which to transfer and adapt that capability.

Both latent and full-text inputs use the adapted decoder, so a reading objective can also change its predictions on ordinary text. ACD therefore adds a full-text anchor that directly regularizes those predictions. Figure 2 summarizes diagnostics of the decoder weights and latent inputs; quantitative measurements and initialization probes are reported in Appendix E.

2.3.1 The Student

The student contains a frozen LCLM encoder, a trainable MLP adapter, and the base decoder augmented with a low-rank reading patch. Decoder normalization gains are also trainable and initialized from the base model; its remaining parameters are frozen.

For each projection W_0, we define $\Delta W = W_{\text{LCLM}} - W_0$ and parameterize the adapted weight as $W_\theta = W_0 + AB$, with $\text{rank}(AB) \leq \rho$. The factors A and B are initialized so that their product equals the rank-ρ truncated singular value decomposition of ΔW. The patch covers all 252 decoder projections. At the default rank $\rho = 64$, it has 132M parameters, and the adapter has 9.2M.

We denote the trainable patch, adapter, and gains by θ, the student by p_θ, and the unmodified teacher by p_0. Teacher predictions are computed before student training and cached. The student uses the same adapted decoder in both views; the compression encoder and adapter are used only when constructing latent inputs.

2.3.2 Two Views, Two Terms

Each trajectory is rendered as a *full-text view* x^{H}, with all observations in hard tokens, and a *latent view* x^{L}, with all observation bodies compressed. Let $\mathcal{J}$ index corresponding assistant tokens and c_j^{H}, c_j^{L} their preceding contexts. Predictions are aligned by assistant-token identity, since compression changes absolute positions.

We also append read-back questions to latent examples. Let $\mathcal{R}$ index their answer tokens, with target y_r and preceding context c_r^{L}. These positions directly supervise recovery of content from compressed observations.

The objective is $\mathcal{L} = \mathcal{L}_{\text{distill}} + \lambda \mathcal{L}_{\text{anchor}}$, where

$$\mathcal{L}_{\text{distill}} = \frac{1}{N_{\text{distill}}} \left[\sum_{j \in \mathcal{J}} D_{\text{KL}}\big(p_0(\cdot \mid c_j^{\text{H}}) \,\big\|\, p_\theta(\cdot \mid c_j^{\text{L}})\big) - \sum_{r \in \mathcal{R}} \log p_\theta(y_r \mid c_r^{\text{L}}) \right], \tag{1}$$

$$\mathcal{L}_{\text{anchor}} = \frac{1}{N_{\text{anchor}}} \sum_{j \in \mathcal{J}} D_{\text{KL}}\big(p_0(\cdot \mid c_j^{\text{H}}) \,\big\|\, p_\theta(\cdot \mid c_j^{\text{H}})\big)\,. \tag{2}$$

Here $N_{\text{distill}} = |\mathcal{J}| + |\mathcal{R}|$ and $N_{\text{anchor}} = |\mathcal{J}|$ count supervised positions within a batch. We use forward KL by default; alternative directions are evaluated in Appendix B.4.

Learning from the latent view. The distillation term encourages the student to make predictions from compressed observations that are consistent with the base model's predictions from the original text. Matching the teacher's distribution provides supervision beyond the single action recorded in the trajectory.

The read-back component adds explicit supervision for extracting literal content from compressed observations. Its answers are substrings of the source observations and account for approximately 3% of supervised positions. The construction and filtering of these questions are described in Appendix D.

Anchoring on the full-text view. The anchor term regularizes the adapted model on inputs that contain no latent observations. Cross-view distillation alone does not directly constrain predictions on these full-text inputs, even though both views use the same adapted decoder.

Because the full-text view bypasses the compression pipeline, the anchor term updates the reading patch and normalization gains but has no gradient with respect to the compression adapter. The latent-view loss updates all three components. Separately normalizing the two terms makes their relative weight explicit through λ, rather than letting it depend on the numbers of supervised tokens in the two views. We use $\lambda = 1$ in all reported runs.

Choice of teacher. Both terms use the original base model as their teacher, but for different purposes: it provides the full-text prediction target for latent reading and the reference distribution for behavioral preservation. Using an external model for the anchor would instead encourage the student to adopt that model's full-text predictions. Self-anchoring directly expresses our objective of retaining the starting agent's behavior while adding latent-reading capability. The distillation teacher need not be identical to the anchor; Appendix B.4 examines their roles separately.

Estimating the divergences. Since the teacher is frozen and the training trajectories are fixed, its predictions can be computed once. At each supervised assistant position, we cache the teacher's top-k_{logits} token log-probabilities, with $k_{\text{logits}} = 64$, together with the remaining probability mass. During training, we evaluate each divergence over the cached tokens and one additional bucket containing the rest of the vocabulary. This coarsened divergence matches the probability mass assigned to the tail but does not constrain its internal distribution. Before numerical safeguards, it is a lower bound on the full-vocabulary KL in either direction. Implementation details and approximation measurements are in Appendices D and E.5.

2.4 Training and Inference

Training. We select the shortest successful CoderForge trajectory per task (Ariyak et al., 2026), yielding 35,140 trajectories; ablations use the first 3,514. Trajectories are rendered in both views and packed into 204,800-token sequences. We train for one epoch with AdamW, using a learning rate of 10^{-4} for the reading patch and normalization gains and 5×10^{-5} for the adapter. Training is offline: predictions are evaluated on recorded trajectory prefixes rather than on interactions generated by the student. Cached targets avoid a teacher forward pass during optimization, while gradients propagate through the student decoder to its trainable components.

A plain-text tool-calling probe checks editing behavior and tool-call validity beyond the training losses. All reported arms are evaluated regardless of probe outcomes. Further optimization and probe details are in Appendix D.

Inference. The adapted agent uses the scaffold's native function-calling interface and ordinary tools. Before each model call, LOHA renders the history with the selected K; observations shorter than 128

characters remain in hard tokens regardless of age. Training thus uses the fully latent and full-text endpoints, whereas mixed-window inference combines older latent observations with recent raw text.

Cost accounting. We distinguish representation length from the cost of an executed trajectory. Encoded observation bodies use approximately one soft token per 16 source tokens. The context-level reduction is smaller because envelopes, short observations, prompts, assistant turns, and the recent window remain in text.

For a fixed trajectory, we compare the compressed representation with the full-text rendering of the same history. Separately, we measure cumulative decoder input over actual rollouts, whose lengths and tool calls may differ across conditions.

Advancing the recent window changes an existing prefix and invalidates its cache from the first modified observation. The current server also re-encodes observations on each call. We therefore report encoder work, prefix reuse, and serving throughput separately from context length (Section 3.4); metric definitions are in Appendix A.

3 EXPERIMENTS

We evaluate task performance and context cost on two base agents, Qwen3-4B-Instruct-2507 and SWE-Master-4B-RL, its RL-fine-tuned derivative. Both use the same LOHA + ACD recipe, with each agent anchored to its own unmodified model. We sweep the recency window for both families; further Qwen3 experiments examine context limits, anchor identity, and serving efficiency.

Setup. We evaluate both model families on 499 instances of SWE-bench Verified (Jimenez et al., 2024) using OpenHands (Wang et al., 2025) 0.62.0. Qwen3 uses a 262K-token context window and a 200-iteration cap; SWE-Master uses 131K tokens and 100 iterations. Conditions within each family share the scaffold and decoding settings, with one attempt per instance per seed. The uncompressed base, observation masking, and the two ACD conditions use two or three seeds; pruning baselines use one run. Full protocols are in Appendix A.

Comparisons. The main comparisons use LOHA + ACD at $K{=}3$, the uncompressed base agent, and the same adapted model using full text ($K{=}\infty$). The latter separates adaptation from the effect of context compression. Baselines are observation masking (Lindenbauer et al., 2025), SWE-Pruner (Wang et al., 2026a), Self-Prune, and LongCodeZip (Shi et al., 2025), all using the same recent-observation window and compression threshold.

Metrics. We report resolve rate, context tokens per call, decoder input tokens per trajectory, and serving throughput. The whole-trajectory compression ratio compares the plaintext equivalent of an agent's own contexts with the tokens it actually reads. Encoder work, prefix-cache reuse, and estimated compute are reported separately; metric definitions and paired-test results are in Appendices A and B.

3.1 FULL-WINDOW RESULTS

At $K{=}3$, LOHA reduces context per call by 43% for Qwen3 and 57% for SWE-Master, with a performance cost in both families (Tables 1 and 2). The recency sweep includes $K{=}8$ as a larger-window alternative (Table 3a).

Qwen3-4B-Instruct. LOHA + ACD resolves 12.1% of instances against 14.5% for the uncompressed base and 13.2% for the adapted full-text agent. Mean context per call falls from the base agent's 43.6K to 24.8K tokens. Masking reaches a similar resolve rate (12.2%) with 20.1K tokens per call. The pruning baselines resolve 13.4–14.2% while retaining 29–39K tokens per call.

SWE-Master-4B-RL. LOHA + ACD resolves 21.8% against 27.5% for the uncompressed base and 24.3% for the adapted full-text agent, while reducing mean context per call from 51.8K to 22.2K tokens. Masking drops to 17.6%; retaining a latent history therefore preserves more task performance on this base. SWE-Pruner and LongCodeZip resolve 24.0%, and Self-Prune matches LOHA's 21.8%, with all three retaining larger contexts (34–49K tokens per call).

Trajectory cost. Per-call savings translate differently across the two agents. Qwen3 with LOHA makes more calls than its base (60.9 versus 37.9), so decoder input falls only from 1.65M to 1.51M

Table 1: Qwen3-4B-Instruct-2507 (262K context, 200 iterations). Resolved: out of 499, mean ± standard deviation; pruners use single runs. Cost definitions: Appendix A.

Condition	resolved	calls	in/call	dec in	cached	enc/aux tok	ratio	dec PF	enc/aux PF
Uncompressed anchor	72.3 ± 3.1 (14.5%)	37.9	43.6K	1.65M	96.9%	—	1.00×	1.82	—
LOHA + ACD (ours)									
plain text ($K{=}\infty$)	66.0 ± 2.8 (13.2%)	48.0	50.9K	2.45M	95.2%	—	1.00×	4.24	—
Hard-Last-3 (default)	60.5 ± 0.7 (12.1%)	60.9	24.8K	1.51M	92.5%	1.86M	2.24×	2.10	1.64
Reduction baselines (same window and threshold)									
Masking ($K{=}3$) (Lindenbauer et al., 2025)	61.0 ± 5.3 (12.2%)	60.6	20.1K	1.22M	91.5%	—	2.39×	1.69	0
SWE-Pruner (Wang et al., 2026a)	67 (13.4%)	48.5	29.1K	1.41M	94.0%	11.8K	1.47×	1.83	0.01
Self-Prune	67 (13.4%)	50.4	38.8K	1.96M	94.3%	17.6K	1.34×	2.72	0.13
LongCodeZip (Shi et al., 2025)	71 (14.2%)	43.5	38.3K	1.67M	96.1%	13.6K	1.01×	1.62	0.04

Table 2: SWE-Master-4B-RL (131K context, 100 iterations). Reporting follows Table 1; each family uses its own uncompressed reference.

Condition	resolved	calls	in/call	dec in	cached	enc/aux tok	ratio	dec PF	enc/aux PF
Uncompressed teacher	137.0 ± 5.7 (27.5%)	85.0	51.8K	4.40M	97.2%	—	1.00×	3.89	—
LOHA + ACD (ours)									
plain text ($K{=}\infty$)	121.5 ± 6.4 (24.3%)	78.9	50.1K	3.95M	92.3%	—	1.00×	6.77	—
Hard-Last-3 (default)	109.0 ± 7.1 (21.8%)	81.5	22.2K	1.81M	85.0%	2.45M	2.35×	4.34	2.16
Masking ($K{=}3$) (Lindenbauer et al., 2025)	88.0 ± 1.0 (17.6%)	86.6	18.9K	1.64M	82.3%	—	2.62×	4.22	0
SWE-Pruner (Wang et al., 2026a)	120 (24.0%)	82.8	36.3K	3.00M	86.9%	57.9K	1.36×	7.75	0.07
Self-Prune	109 (21.8%)	76.4	33.8K	2.58M	92.0%	78.8K	1.47×	4.13	0.57
LongCodeZip (Shi et al., 2025)	120 (24.0%)	79.3	48.6K	3.85M	91.5%	66.6K	1.00×	7.60	0.17

tokens per trajectory. SWE-Master makes a similar number of calls (81.5 versus 85.0), and decoder input falls from 4.40M to 1.81M. The representation ratios, measured against each agent's own contexts rendered in full text, are $2.24\times$ and $2.35\times$, respectively.

3.2 Performance under Context Limits

For Qwen3, we repeat the comparison with 64K- and 32K-token context limits on a 199-instance subset. A trajectory that exceeds the limit is stopped and its patch is graded. At 64K, the four conditions resolve 31–36 instances. At 32K, Hard-Last-3 resolves 42, compared with 22 for the adapted full-text agent, 26 for masking, and 31 for the base agent (Figure 3a). The gains over the adapted full-text agent and masking are supported by paired McNemar tests ($p{=}0.0002$ and $p{=}0.0025$, respectively).

Masking exceeds the 32K limit less often than LOHA (23 versus 36 trajectories) but resolves fewer tasks (Figure 3b). Fitting within the context window is therefore only part of the benefit: the results also favor retaining a compressed history over deleting older observations. Full counts and paired comparisons are in Appendix B.2.

3.3 Ablation Studies

We vary the recency window for both model families and the distillation teacher for Qwen3. The teacher ablation uses rank-64 students trained on 3,514 trajectories.

Recency-Window Sweep. We sweep $K \in \{0, 1, 2, 3, 4, 8\}$ and full text while holding each trained student fixed (Table 3a). Larger windows generally improve resolve rate while retaining more observations in text. Both models lose performance at $K{=}1$ and again at $K{=}0$; a separate 20-task diagnostic finds that increasing patch rank does not repair the fully latent policy (Appendix B.3).

In this single-run sweep, increasing K from 3 to 8 raises resolved counts from 68 to 72 on Qwen3 (13.6% to 14.4%) and from 111 to 115 on SWE-Master (22.2% to 23.0%). Estimated decoder compute rises from 1.67 to 2.10 PF and from 3.74 to 8.37 PF per trajectory, respectively. The trend is not strictly monotonic: Qwen3 resolves 76 tasks at $K{=}4$. Thus, $K{=}3$ is a compact default, while $K{=}8$ offers higher resolve rates at greater text retention and compute cost.

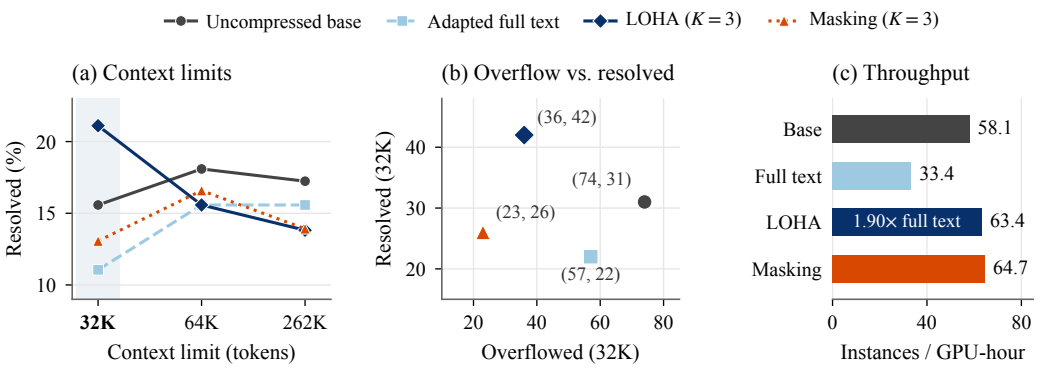


Figure 3: Qwen3: (a) resolve rates on 199 instances: single runs at 32K/64K, seed means at 262K. (b) Overflow and resolved counts at 32K. (c) Single-GPU throughput at concurrency 16. Protocols and paired tests: Appendix B.

Table 3: Recency and teacher ablations. (a) 499 instances per model; dec PF: estimated decoder PFLOPs per trajectory. (b) 199 random instances with the teacher supplying both targets, followed by a separate 20-task diagnostic listing teacher/anchor pairs. Res.: resolved; patch: non-empty patches; stuck: loop terminations. Single runs except the 199-task base mean. Details: Appendices C and B.4.

(a) Recency-window sweep

	Qwen3		SWE-Master	
K	res.	dec PF	res.	dec PF
∞	80	1.47	128	2.99
8	72	2.10	115	8.37
4	76	2.08	108	4.49
3	68	1.67	111	3.74
2	65	1.40	103	2.93
1	47	1.21	88	1.89
0	37	1.18	51	0.93

(b) Distillation teacher

	Full text			Hard-Last-3		
Teacher	res.	patch	stuck	res.	patch	stuck
Uncompressed base	34.3	—	—	—	—	—
Self (Qwen3-4B)	**28**	140	43	**29**	127	50
Qwen3-30B-A3B	15	68	136	14	59	141
SWE-Master-4B-RL	0	22	59	0	22	68
20 tasks: distillation teacher / anchor						
Uncompressed base	10/20	14	2	—	—	—
Self / self	11/20	14	3	11/20	16	5
30B / 30B	7/20	8	10	6/20	8	9
30B / self	12/20	15	5	9/20	13	6

On 358 literals quoted by later actions, the released LCLM decoder achieves exact-recitation rates of 20.4%, 15.6%, and 15.1% at $4\times$, $8\times$, and $16\times$ compression, respectively (Appendix E.6). The low fidelity even at $4\times$ supports retaining recent source text for exact tool arguments.

Distillation Teacher. The 199-task block of Table 3b uses the distillation teacher for both targets. Under full text and Hard-Last-3, the self-teacher resolves 28 and 29 tasks; the 30B teacher resolves 15 and 14 with more loop terminations, and SWE-Master resolves none. On the same 20 diagnostic tasks, the rank-64 self-teacher resolves 11 tasks under either policy; using the 30B model for both targets resolves seven and six. Keeping the base anchor with the 30B distillation teacher gives 12 and nine, with forward KL for both terms. The latter uses top-64 rather than top-16 cached targets. These diagnostics support retaining the base anchor while allowing a different distillation teacher.

Better literal recall need not improve task performance. At rank 256, the student using the 30B model for both targets exactly recites 35/96 literals, versus 26/96 for the self-anchored student, but resolves only 4/20 tasks versus 10/20 under either context policy. Full controls and KL-direction ablations are in Appendix B.4.

Reading Initialization. On Qwen3, the rank-64 initialization retains 17% of the LCLM–base weight-difference energy yet recovers 79% of the LCLM decoder's structured read-back score. Before ACD training, this patch still degrades agent behavior (Appendix E).

3.4 SERVING EFFICIENCY

For Qwen3, on one GPU with 16 concurrent trajectories, LOHA completes 63.4 instances per GPU-hour, compared with 33.4 for the adapted full-text agent and 58.1 for the base agent (Figure 3c). This is a $1.9\times$ throughput increase over the adapted full-text condition, with median call latency falling from 5.07 to 1.71 seconds and 90th-percentile latency from 45.6 to 7.17 seconds. Masking reaches 64.7 instances per GPU-hour. At the measured mean context lengths, the shared 416K-token KV cache accommodates about 11 adapted full-text trajectories versus 20 with Hard-Last-3, below and above the concurrency of 16, respectively. Full serving measurements are in Appendix B.5.

In the current server, Qwen3 with LOHA uses an estimated 2.10 PF of decoder compute and 1.64 PF of encoder compute per trajectory, against 1.82 PF for the base agent. The server re-encodes old observations at every call. A replay with per-observation caching reduces the encoder contribution to 0.02 PF and estimates total compute at 2.12 PF per trajectory. For SWE-Master, estimated compute is 4.34 PF in the decoder and 2.16 PF in the encoder, against 3.89 PF for its uncompressed base.

4 RELATED WORK

Context and code compression. Agent histories can be shortened by masking (Lindenbauer et al., 2025), summarization (Packer et al., 2023; Kang et al., 2026), or folding (Sun et al., 2025; Ye et al., 2025). SWE-Pruner and SWE-Pruner Pro select observation lines (Wang et al., 2026a;b); CoACT uses action-preserving targets (Chen et al., 2026), while TACO evolves output-filtering rules (Ren et al., 2026). Prompt compressors select tokens or spans (Li et al., 2023; Jiang et al., 2023; Pan et al., 2024), and LongCodeZip and SlimCode exploit code structure (Shi et al., 2025; Wang et al., 2024). LOHA instead keeps older observations in latent form and recent content verbatim. Masking provides a direct comparison by using the same recency window without retaining a latent history.

Soft-token compression and memory. Gist tokens, in-context autoencoders, and extreme compressors establish the soft-token interface (Mu et al., 2023; Ge et al., 2024; Li et al., 2025); LCLM scales it to long documents (Li et al., 2026). Latent memory systems target experience reuse and agent coordination (Zhang et al., 2026; 2025; Fu et al., 2026). Our setting instead requires compressed observations to support repeated tool interactions within a trajectory, including actions that need exact text.

Distillation and anchoring. Context distillation transfers predictions from richer to reduced inputs (Askell et al., 2021; Snell et al., 2022; Cheng et al., 2024). Behavioral anchoring relates to Learning without Forgetting (Li & Hoiem, 2018), self-distillation (Yang et al., 2024b), and reference-policy KL regularization (Ziegler et al., 2019; Ouyang et al., 2022). ACD combines these roles: the same base model supplies the latent-view target and full-text reference, while a low-rank patch (Hu et al., 2022) enables adaptation. Our ablations distinguish the effects of the distillation teacher and behavioral anchor. Extended comparisons are in Appendix F.

5 CONCLUSION AND FUTURE WORKS

LOHA and ACD combine observation compression with behavior-preserving adaptation for software-engineering agents. LOHA keeps recent observations and agent turns in text while compressing older observations; ACD teaches latent reading while anchoring full-text behavior to the base policy. At $K{=}3$, context per call falls by 43% on Qwen3 and 57% on SWE-Master. Larger recency windows trade compression for task performance, while the 32K experiments demonstrate the value of retaining compressed history. The teacher ablations further motivate treating context representation and policy adaptation as a joint design problem.

Future work can pursue three directions. First, learning to allocate observation fidelity by anticipated tool use could jointly determine what to compress and when to retrieve exact source text. Second, objectives that constrain tool-use decisions or trajectory-level outcomes, rather than only next-token distributions, could offer new ways to balance latent-reading gains against policy drift. Third, training on the adapted agent's own trajectories could couple representation learning to the actions and observations it induces, with task success guiding what information the compressed history preserves. Together, these directions would connect compression more closely to how an agent uses its context.

REPRODUCIBILITY STATEMENT

We will release the code and trained model weights.

AI ASSISTANCE

The method, hypotheses, loss functions, and experimental design were developed by the authors. Generative AI tools only assisted with literature discovery, drafting and polishing text, and writing or debugging code for dataset processing, training, inference, and evaluation. The authors reviewed the full manuscript, all code, and the original experimental logs and statistical results, tested the code, and took responsibility for all content and artifacts.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *Proceedings of the International Conference on Learning Representations*, 2024.

Alpay Ariyak, Junda Zhang, Junxiong Wang, Shang Zhu, Federico Bianchi, Sanjana Srivastava, Ashwinee Panda, Siddhant Bharti, Chenfeng Xu, John Heo, Xiaoxia Shirley Wu, James Zhou, Percy Liang, Leon Song, Ce Zhang, Ben Athiwaratkun, Zhongzhu Zhou, and Qingyang Wu. CoderForge-Preview: SOTA open dataset for training efficient agents. `https://huggingface.co/datasets/togethercomputer/CoderForge-Preview`, 2026.

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. A general language assistant as a laboratory for alignment. arXiv preprint arXiv:2112.00861, 2021.

Haorui Chen, Yuancheng Zhu, Yitong Zhang, and Jia Li. CoACT: Action-preserving observation compression for coding agents. arXiv preprint arXiv:2607.02911, 2026.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xRAG: Extreme context compression for retrieval-augmented generation with one token. In *Advances in Neural Information Processing Systems*, 2024.

Muxin Fu, Xiangyuan Xue, Yafu Li, Zefeng He, Siyuan Huang, Xiaoye Qu, Yu Cheng, and Yang Yang. LatentMem: Customizing latent memory for multi-agent systems. arXiv preprint arXiv:2602.03036, 2026.

Tommaso Furlanello, Zachary C. Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *Proceedings of the International Conference on Machine Learning*, pp. 1607–1616, 2018.

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 *Proceedings of the International Conference on Learning Representations*, 2024.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *Proceedings of the International Conference on Learning Representations*, 2024.

Arnav Gudibande, Eric Wallace, Charlie Snell, Xinyang Geng, Hao Liu, Pieter Abbeel, Sergey Levine, and Dawn Song. The false promise of imitating proprietary language models. In *Proceedings of the International Conference on Learning Representations*, 2024.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 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 *Proceedings of the International Conference on Learning Representations*, 2022.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *Proceedings of the International Conference on Learning Representations*, 2024.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A. Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. ACON: Optimizing context compression for long-horizon LLM agents. In *Proceedings of the International Conference on Machine Learning*, 2026.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114 (13):3521–3526, 2017.

Ang Li, Sean McLeish, Haozhe Chen, Nimit Kalra, Zaiqian Chen, Artem Gazizov, Venkata Anoop Suhas Kumar Morisetty, Bhavya Kailkhura, Harshitha Menon, Zhuang Liu, et al. End-to-end context compression at scale. arXiv preprint arXiv:2606.09659, 2026.

Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. Compressing context to enhance inference efficiency of large language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018.

Zongqian Li, Yixuan Su, and Nigel Collier. 500xCompressor: Generalized prompt compression for large language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 25081–25091, 2025.

Tobias Lindenbauer, Igor Slinko, Ludwig Felder, Egor Bogomolov, and Yaroslav Zharov. The complexity trap: Simple observation masking is as efficient as LLM summarization for agent context management. arXiv preprint arXiv:2508.21433, 2025.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020.

Jesse Mu, Xiang Lisa Li, and Noah Goodman. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems*, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. MemGPT: Towards LLMs as operating systems. arXiv preprint arXiv:2310.08560, 2023.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics: ACL*, 2024.

Qwen Team. Qwen3 technical report. arXiv preprint arXiv:2505.09388, 2025.

Jincheng Ren, Siwei Wu, Yizhi Li, Kang Zhu, Shu Xu, Boyu Feng, Ruibin Yuan, Wei Zhang, Riza Batista-Navarro, Jian Yang, and Chenghua Lin. A self-evolving framework for efficient terminal agents via observational context compression. arXiv preprint arXiv:2604.19572, 2026.

Yuling Shi, Yichun Qian, Hongyu Zhang, Beijun Shen, and Xiaodong Gu. LongCodeZip: Compress long context for code language models. In *Proceedings of the IEEE/ACM International Conference on Automated Software Engineering*, 2025.

Charlie Snell, Dan Klein, and Ruiqi Zhong. Learning by distilling context. arXiv preprint arXiv:2209.15189, 2022.

Weiwei Sun, Miao Lu, Zhan Ling, Kang Liu, Xuesong Yao, Yiming Yang, and Jiecao Chen. Scaling long-horizon LLM agent via context-folding. arXiv preprint arXiv:2510.11967, 2025.

Xingyao Wang, Boxuan Li, Yufan Song, Frank F. Xu, Xiangru Tang, Mingchen Zhuge, Jiayi Pan, Yueqi Song, Bowen Li, Jaskirat Singh, et al. OpenHands: An open platform for AI software developers as generalist agents. In *Proceedings of the International Conference on Learning Representations*, 2025.

Yan Wang, Xiaoning Li, Tien N. Nguyen, Shaohua Wang, Chao Ni, and Ling Ding. Natural is the best: Model-agnostic code simplification for pre-trained large language models. *Proceedings of the ACM on Software Engineering*, 1(FSE):586–608, 2024. doi: 10.1145/3643753.

Yuhang Wang, Yuling Shi, Mo Yang, Rongrui Zhang, Shilin He, Heng Lian, Yuting Chen, Siyu Ye, Kai Cai, and Xiaodong Gu. SWE-Pruner: Self-adaptive context pruning for coding agents. arXiv preprint arXiv:2601.16746, 2026a.

Yuhang Wang, Yuling Shi, Shaoqiu Zhang, Jialiang Liang, Shilin He, Siyu Ye, Yuting Chen, Kai Cai, and Xiaodong Gu. SWE-Pruner Pro: The coder LLM already knows what to prune. arXiv preprint arXiv:2607.18213, 2026b.

Yating Wu, Yuhao Zhang, Sayan Ghosh, Sourya Basu, Anoop Deoras, Jun Huan, and Gaurav Gupta. ContextWeaver: Selective and dependency-structured memory construction for LLM agents. arXiv preprint arXiv:2604.23069, 2026.

Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying LLM-based software engineering agents. *Proceedings of the ACM on Software Engineering*, 2(FSE):801–824, 2025. doi: 10.1145/3715754.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, 2024a.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling data for software engineering agents. In *Advances in Neural Information Processing Systems*, pp. 107402–107452, 2025.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. Self-distillation bridges distribution gap in language model fine-tuning. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 2024b.

Rui Ye, Zhongwang Zhang, Kuan Li, Huifeng Yin, Zhengwei Tao, Yida Zhao, Liangcai Su, Liwen Zhang, Zile Qiao, Xinyu Wang, et al. AgentFold: Long-horizon web agents with proactive context management. arXiv preprint arXiv:2510.24699, 2025.

Guibin Zhang, Muxin Fu, and Shuicheng Yan. MemGen: Weaving generative latent memory for self-evolving agents. arXiv preprint arXiv:2509.24704, 2025.

Guibin Zhang, Yaohui Ling, Fanci Meng, Kun Wang, and Shuicheng Yan. Mem-W: Latent memory-native GUI agents. arXiv preprint arXiv:2605.09317, 2026.

Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. AutoCodeRover: Autonomous program improvement. In *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2024.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences. arXiv preprint arXiv:1909.08593, 2019.

A EXPERIMENTAL PROTOCOL

The main comparison evaluates LOHA + ACD and the same compression baselines on Qwen3-4B-Instruct-2507 and SWE-Master-4B-RL. This section specifies the common scaffold, each model family's protocol, and the reported metrics. Additional results are in Appendix B; implementation and training details are in Appendix D.

A.1 BENCHMARK, AGENTS, AND DECODING

We evaluate on 499 of the 500 SWE-bench Verified instances (Jimenez et al., 2024). The instance `django__django-12273` is excluded from every condition because its official evaluation image fails during container initialization (`/usr/bin/cat` is missing, exit code 127).

Both model families use OpenHands (Wang et al., 2025) 0.62.0 with CodeActAgent and native function calling. The tools are `execute_bash`, `str_replace_editor`, `think`, `finish`, and `task_tracker`; browsing, Jupyter, and the history condenser are disabled. Qwen3 uses a 262,144-token context window and a 200-iteration cap. SWE-Master uses its native tool rendering, a 131,072-token window, and a 100-iteration cap. Conditions within each family share decoding settings; for Qwen3 these are temperature 0.7, top-p 0.8, and top-k 20, enforced in both the server and harness.

Each instance contributes one final result per seed. The main evaluation uses a 7,200-second wall-clock limit; an in-process alarm at 6,600 seconds ends the trajectory and extracts its patch. We run three seeds for the Qwen3 base and masking, and two for its ACD conditions. For SWE-Master, we run two seeds for the base and ACD conditions and three for masking. Pruning baselines use one run per family. Per-seed counts and recency-window sweeps are listed in Appendix C.

A.2 CONDITIONS AND BASELINES

Each model family has its own uncompressed reference. The ACD student is initialized from that model and uses it as the teacher for both objectives. Comparing the student in full text with the base measures adaptation; comparing the same student in full text and under LOHA measures the change in context representation.

- **Uncompressed base.** The original Qwen3 or SWE-Master agent with full-text observations.
- **LOHA + ACD.** The rank-64 self-anchored student, evaluated with Hard-Last-3 and full text ($K{=}\infty$) in the main comparisons. Recency sweeps test $K \in \{0, 1, 2, 3, 4, 8\}$ for both model families; the 20-instance diagnostic also tests rank 256.
- **Observation masking** (Lindenbauer et al., 2025). The last three observations remain in text, and older observations are removed. This shares LOHA's recent text window and tests the value of retaining a latent history.
- **SWE-Pruner** (Wang et al., 2026a). A 0.6B reranker selects observation lines using a focus question derived from the agent's last action.
- **Self-Prune.** The 4B decoder selects numbered lines through guided JSON, with at most 400 lines per pruning call. It uses the same interface as SWE-Pruner without a separate model.
- **LongCodeZip** (Shi et al., 2025). A 1.5B ranker selects code chunks and compresses observations in token space.

All compression baselines use the same $K{=}3$ recent-observation window and 128-character threshold. The pruning servers cache the compressed text and process each observation once. The current LOHA server re-encodes compressed observations at each model call; the measured cost and a replay with encoder caching are reported separately in Appendix B.5.

A.3 SCAFFOLD CHANGES AND GRADING

The shared scaffold fork renders observations according to the selected context policy. Observations shorter than 128 characters remain in text under every inference policy. The editor also supports a bounded fallback when a `str_replace` argument does not match exactly: it first normalizes

whitespace line by line, then tries a semi-global edit-distance match accepted only when unique by a margin. A syntax check precedes writing. Fallback events are logged. The Qwen3 comparisons also share three editor hints for unchanged files, missing parameters, and unresolved paths.

Containers run rootless on the official SWE-bench images through `udocker`'s `fakechroot` engine. Host paths can appear in observation output because the engine does not rewrite those strings; the inspected tool arguments address `/workspace/...`. This behavior is shared across conditions.

Grading uses the official SWE-bench harness, version 4.1.0. We read its complete log from the container file system because truncation in the scaffold's observation layer can remove the start-of-test marker and misclassify a passing patch as unapplied. Gold patches were used to validate this grading path. Empty predictions count as unresolved.

A.4 METRICS

Task performance and behavior. Resolve rate is the fraction of the 499 instances that pass the benchmark grader. In the ablation tables, *res.* is the resolved count, *patch* counts non-empty patches touching a file other than the build manifest, *stuck* counts loop terminations, and *cap* counts iteration-limit terminations. The *edits* and *calls* columns are mean `str_replace` edits and model calls per trajectory. We also log tool-call validity and editor-fallback use.

Token cost. The *in/call* column is mean decoder prompt length; *dec in* sums prompt tokens over a trajectory; *cached* is the share served from the prefix cache. The *enc/aux tok* column separately counts tokens processed by the observation encoder or baseline compressor. Self-Prune's auxiliary work uses the decoder itself.

The *ratio* column divides the plaintext equivalent of an agent's own contexts by the decoder tokens actually read. It measures the representation reduction for those contexts. Actual token savings between agents also depend on the trajectories they generate. Encoded observation bodies have a $16\times$ ratio, while prompts, assistant turns, tool envelopes, and recent observations remain in text.

Compute and serving. We estimate compute per trajectory in petaflops (*PF*), using $2P$ FLOPs per computed token plus the quadratic attention term, with cached positions excluded from prefill. The decoder has P=3.63B non-embedding parameters; encoder and auxiliary models use dense-only estimates at their respective sizes. Serving measurements report completed instances per GPU-hour, trajectory wall time, and per-call latency. Their workload and completion cutoff are given in Appendix B.5.

Reading probes. We measure exact recitation of 96 literal fields and structured extraction of 48 fields from compressed tool outputs. The structured probe uses exact-JSON and substring scoring to distinguish formatting errors from content recovery. These adaptation probes are separate from the 358-literal compression study in Appendix E.6.

B ADDITIONAL RESULTS

This section supplements the two main result tables with paired comparisons, context-limit experiments, ablations, and serving measurements. Unless stated otherwise, the ablations and serving experiments use the Qwen3 family.

B.1 FULL-WINDOW COMPARISONS

Qwen3-4B-Instruct. For Table 1, paired McNemar tests against the base agent give p=0.19 and 1.00 for the adapted full-text condition, and p=0.049 and 0.32 for Hard-Last-3. In the first Hard-Last-3 comparison, 18 instances are resolved only by the adapted agent and 33 only by the base. The two Hard-Last-3 seeds have representation ratios of $2.20\times$ and $2.27\times$.

Masking has the largest variation across seeds and averages 99 iteration-cap terminations against 43 for the base. The observation compression ratios are $1.02\times$ for LongCodeZip, $2.9\times$ for SWE-Pruner, and $1.8\times$ for Self-Prune, compared with LOHA's $16\times$. LongCodeZip retains nearly every chunk in its coarse ranking stage on these observations.

Table 4: Resolve rate under a capped context window on a 199-instance subset, one run per cell; the 262K column is the mean over the seeds of Table 1 on the same subset. *ovf*: trajectories stopped at the cap. Paired tests in the text.

	262K	**64K**		**32K**	
Condition	res.	res.	ovf	res.	ovf
Uncompressed anchor	34.3/199	36/199	24	31/199	74
LOHA plain text ($K{=}\infty$)	31.0/199	31/199	25	22/199	57
LOHA Hard-Last-3	27.5/199	31/199	13	42/199	36
Masking ($K{=}3$)	27.7/199	33/199	7	26/199	23

Table 5: Behavior and cost of the self-anchored model under the three settings of the recency policy, on the 20-instance subset of Table 6. *dec tok*, *enc tok*: decoder prompt and encoder input tokens per trajectory; other columns as in Appendix A.

model	**policy**	res.	patch	stuck	cap	dec tok	cached	enc tok	ratio
Uncompressed agent	plain	10/20	14	2	1	846K	98.0%	—	1.00×
ACD, $\rho{=}64$	$K{=}\infty$	11/20	14	3	4	2.39M	99.2%	—	1.00×
ACD, $\rho{=}64$	$K{=}3$	**11/20**	16	5	1	632K	92.1%	763K	2.21×
ACD, $\rho{=}64$	$K{=}0$	1/20	4	5	9	2.30M	99.6%	4.10M	2.78×
ACD, $\rho{=}256$	$K{=}\infty$	10/20	15	3	1	1.74M	98.4%	—	1.00×
ACD, $\rho{=}256$	$K{=}3$	10/20	16	2	2	505K	95.4%	188K	1.37×
ACD, $\rho{=}256$	$K{=}0$	2/20	3	5	8	3.17M	99.7%	6.98M	3.20×

SWE-Master-4B-RL. For Table 2, paired tests against the SWE-Master base give $p{=}0.13$ and 0.12 for the adapted full-text condition, $p{=}0.07$ and $p{<}0.001$ for Hard-Last-3, and $p{<}0.001$ in both comparisons for which masking and base runs are paired. The mean resolve-rate gaps are 3.2 points for adapted full text, 5.7 for Hard-Last-3, and 9.9 for masking.

SWE-Pruner and LongCodeZip each resolve 120 instances, and Self-Prune resolves 109, compared with a mean of 109 for LOHA. Their contexts remain larger: 34–49K tokens per call against LOHA's 22.2K. Measured observation ratios are $1.7\times$, $1.9\times$, and $1.01\times$ for SWE-Pruner, Self-Prune, and LongCodeZip, respectively. All comparisons in this family use SWE-Master as the starting agent and, for ACD, as its own anchor.

B.2 Context-Limit Experiments

Table 4 evaluates the four Qwen3 conditions on the same 199-instance subset with context limits of 65,536 and 32,768 tokens, reserving 4,096 tokens for generation. Each capped condition is run once. The 262K column is the mean from the full-window seeds on this subset. When a trajectory exceeds the limit, generation stops and the current patch is graded.

At 32K, the paired comparison between Hard-Last-3 and adapted full text has 24 instances resolved only by Hard-Last-3 and four only by full text ($p{=}0.0002$). Against masking the counts are 21 and five ($p{=}0.0025$); against the base they are 19 and eight ($p{=}0.052$). Of the overflowed trajectories, four base runs, zero adapted full-text runs, one Hard-Last-3 run, and two masking runs still yield passing patches. Masking overflows less often than Hard-Last-3 but resolves fewer tasks, supporting the value of retaining compressed history.

B.3 Recency Ablation

The 20-instance recency diagnostic and the teacher and objective ablations use Qwen3 students trained on the first 3,514 trajectories in a fixed corpus ordering. The diagnostic subset contains 20 instances drawn from the 69 solved by one full-benchmark Qwen3 base run. The base solves 10 of these on re-evaluation. This subset is used to examine behavioral failures, and its resolve rates are reported separately from the main benchmark results.

Table 6: Anchor ablation. Each row is one ACD run, varying the model from which both terms distill and the patch rank ρ, evaluated in plain text ($K{=}\infty$) and under Hard-Last-3. Upper panel: the 20-instance subset described in the text; SWE-Master rows were stopped early at zero patches and show their n. Lower panel: the three $\rho{=}64$ students on a 199-instance subset, one run each, with the untrained agent's seed mean as reference. Column definitions in Appendix A.

		plain text ($K{=}\infty$)					**Hard-Last-3**				
anchor	ρ	res.	patch	stuck	edits	calls	res.	patch	stuck	edits	calls
Uncompressed agent (no patch)	—	10/20	14	2	4.8	23.7			—		
itself (Qwen3-4B-Instruct)	64	**11/20**	14	3	8.5	51.1	**11/20**	16	5	6.8	30.5
itself	256	10/20	15	3	4.1	25.4	10/20	16	2	1.5	30.8
Qwen3-30B-A3B (same family, larger)	64	7/20	8	10	0.4	65.0	6/20	8	9	6.3	67.9
Qwen3-30B-A3B	256	4/20	5	15	1.0	41.4	4/20	6	12	0.8	50.8
SWE-Master-4B-RL (fine-tuned agent)	64	0/8	0	7	0.0	49.0	0/5	0	5	0.0	30.6
SWE-Master-4B-RL	256	0/5	0	4	0.0	116.2	0/2	0	2	0.0	6.5
Same three $\rho{=}64$ students on a 199-instance subset (seed-0 shuffle of Verified; cap = 200-iteration cap)											
anchor	ρ	res.	patch	stuck	cap	calls	res.	patch	stuck	cap	calls
Uncompressed agent (no patch)	—	34.3/199	—	—	—	—			—		
itself (Qwen3-4B-Instruct)	64	28/199	140	43	24	43.2	29/199	127	50	34	54.3
Qwen3-30B-A3B	64	15/199	68	136	17	36.5	14/199	59	141	22	44.0
SWE-Master-4B-RL	64	0/199	22	59	15	24.2	0/199	22	68	7	20.6

At rank 64, Hard-Last-3 and full text both resolve 11 tasks; at rank 256, both resolve 10. Hard-Last-3 produces 16 non-empty patches at either rank, compared with 14 and 15 for full text. Fully latent observations reduce these counts to one or two resolved tasks and three or four patches.

The fully latent agents remain active, averaging 18–26 edits and 94–103 model calls per trajectory, with eight or nine runs reaching the iteration cap. Their trajectories include reconstructed edit strings, inaccurate test commands, and repeated views that return another compressed observation. Keeping recent observations in text addresses this failure mode on the diagnostic subset.

B.4 Anchor and Objective Ablations

Reference model and patch rank. Table 6 varies the reference for both ACD objectives and the patch rank. The upper panel uses the 20-instance subset above. The external SWE-Master-anchor runs were stopped early after producing no patches; their completed sample sizes are shown. The lower panel evaluates the three rank-64 students on a 199-instance random subset, selected by a seed-0 shuffle, with one run per condition.

Self-anchored students retain task performance at both ranks on the small subset. Using the 30B reference increases loop terminations and reduces patch production; increasing rank from 64 to 256 does not repair this behavior. The larger subset confirms the ordering: self-anchoring resolves 28 tasks in full text and 29 under Hard-Last-3, the 30B reference resolves 15 and 14, and the external SWE-Master reference resolves none. The SWE-Master-reference runs produce 22 patches under each policy and reach the two-hour limit in 46 of 398 trajectories.

Reading scores do not follow agent performance. A rank-256 student using the 30B reference recites 35 of 96 literals exactly, compared with 26 for a self-anchored student, despite its worse benchmark behavior. The self-anchored subset-trained model also has resolved counts close to the full-data model on the 199-instance subset: 28 versus 31.0 in full text, and 29 versus 27.5 under Hard-Last-3.

Distillation teacher and KL direction. Table 7 keeps the behavioral anchor at the base model while varying the distillation teacher and each KL direction. The seven variants use top-64 cached teacher distributions; the reference row uses top-16 distributions. All use rank 64 and the same 20-instance subset.

Six variants resolve 8–12 tasks in full text and 6–10 under Hard-Last-3. The remaining variant, a forward-KL anchor with reverse-KL distillation from the 30B teacher, resolves five in full text and has 10 loop terminations. Fully latent evaluation resolves only 1–3 tasks in every variant. There is

Table 7: Direction of each divergence and teacher of the distillation term, with the anchor fixed to the base model (ρ=64; subset and columns as in Table 6). F: forward KL; R: reverse KL. The reference row is the self-anchored run of Table 6.

			plain text (K=∞)					Hard-Last-3				
anchor KL	**distill. teacher**	**distill. KL**	res.	patch	stuck	edits	calls	res.	patch	stuck	edits	calls
F	itself (4B)	F (ref.; top-16)	11/20	14	3	8.5	51.1	11/20	16	5	6.8	30.5
F	itself (4B)	R	10/20	13	4	4.4	40.8	9/20	12	4	17.0	68.8
R	itself (4B)	F	11/20	13	4	9.1	34.3	6/20	10	9	9.2	44.5
R	itself (4B)	R	9/20	14	3	8.6	42.6	8/20	13	8	1.3	13.9
F	Qwen3-30B-A3B	F	12/20	15	5	7.1	37.7	9/20	13	6	1.9	37.6
F	Qwen3-30B-A3B	R	5/20	8	10	3.9	23.2	7/20	10	9	4.5	48.4
R	Qwen3-30B-A3B	F	8/20	13	7	5.4	27.2	8/20	14	5	4.0	50.2
R	Qwen3-30B-A3B	R	10/20	14	6	4.4	23.6	10/20	12	7	8.9	43.4

Table 8: Serving throughput on one GPU per condition (RTX PRO 6000, 96 GB), 100 instances at 16 concurrent trajectories. *inst/GPU-h*: completions per GPU-hour up to the 84th, before the tail of long trajectories; *wall*: trajectory wall-clock, median and 90th percentile (minutes); *latency*: per-call decoder latency, median and 90th percentile (seconds); *KV*: KV-cache capacity in tokens; *adm.*: full-context trajectories it admits at in/call.

Condition	**inst/GPU-h**	**wall (min)**	**latency (s)**	**in/call**	**KV**	**adm.**
Uncompressed anchor	58.1	9.1 / 21.0	2.04 / 9.99	35.6K	547K	15.4
LOHA plain text (K=∞)	33.4	9.4 / 85.4	5.07 / 45.6	38.1K	416K	10.9
LOHA Hard-Last-3	63.4	8.5 / 17.7	1.71 / 7.17	20.8K	416K	20.0
Masking (K=3)	64.7	8.3 / 20.4	1.44 / 7.13	19.9K	416K	20.9

no consistent advantage from reversing either KL direction or using the larger distillation teacher. Compared with replacing both references, retaining the base-model anchor avoids the widespread collapse in most variants.

Cross-entropy variant. An earlier experiment replaced latent-view distillation with cross-entropy on trajectory tokens while keeping the anchor loss. Under a preliminary harness, this variant improved structured extraction from 17 to 24 fields and exact recitation from 22 to 28, but edited no files across eight plain-text tasks, emitted 15 malformed tool calls, and never called `str_replace`. The complete objective solved five of those eight tasks. This preliminary result illustrates why the behavioral probe accompanies the reading probes.

B.5 Serving Measurements and Encoder Cost

Workload. Table 8 uses one RTX PRO 6000 GPU with 96 GB of memory per condition, the same 100 Qwen3 evaluation instances, and 16 concurrent trajectories. Throughput is measured through the 84th completion to exclude the final tail of long trajectories. Wall time and latency columns report the median and 90th percentile.

The serving shim keeps the encoder resident and provides 416K tokens of KV-cache capacity, compared with 547K for the base agent served by plain vLLM. At the measured mean context lengths, this accommodates about 20 Hard-Last-3 trajectories, 11 adapted full-text trajectories, and 15 base-agent trajectories. The adapted full-text condition exceeds that capacity at concurrency 16 and has a long latency tail. Masking and LOHA have similar context sizes and throughput.

Prefix caching. In the 20-instance recency ablation, the fully latent layout reuses 99.6–99.7% of decoder prompt tokens from the prefix cache, full text reuses 98.4–99.2%, and Hard-Last-3 reuses 92.1–95.4%. The main Qwen3 and SWE-Master tables report Hard-Last-3 cache shares of 92.5% and 85.0%, respectively. These are measurements of their respective workloads; the cost of advancing the recent window depends on the trajectories.

Encoder replay. The current server encodes every compressed observation again at each call. We replay Qwen3 Hard-Last-3 trajectories to compare this implementation with a cache keyed by

Table 9: Per-seed resolved counts for Tables 1 and 2, with additional reference runs. Counts are out of 499 unless noted; means and standard deviations use the main-evaluation seeds.

Family / condition	additional run	main runs, per seed	mean ± sd
Qwen3-4B-Instruct-2507 family (Table 1)			
Uncompressed anchor	—	73 / 75 / 69	72.3 ± 3.1 (14.5%)
LOHA plain text	80	64 / 68	66.0 ± 2.8 (13.2%)
LOHA Hard-Last-3	68	60 / 61	60.5 ± 0.7 (12.1%)
Masking (K=3)	—	63 / 65 / 55	61.0 ± 5.3 (12.2%)
SWE-Master-4B-RL family (Table 2)			
Uncompressed teacher	113 (of 494)	133 / 141	137.0 ± 5.7 (27.5%)
LOHA plain text	128	117 / 126	121.5 ± 6.4 (24.3%)
LOHA Hard-Last-3	111	114 / 104	109.0 ± 7.1 (21.8%)
Masking (K=3)	—	88 / 87 / 89	88.0 ± 1.0 (17.6%)

Table 10: Recency-window sweeps, one run per condition. Left: Qwen3-4B-Instruct-2507, 200-iteration cap, 262K context. Right: SWE-Master-4B-RL, 100-iteration cap, 131K context. Resolved counts are out of 499; dec PF is estimated decoder compute per trajectory.

	Qwen3-4B family		SWE-Master family	
Policy	res.	dec PF	res.	dec PF
plain text (K=∞)	80	1.47	128	2.99
Hard-Last-8	72	2.10	115	8.37
Hard-Last-4	76	2.08	108	4.49
Hard-Last-3	68	1.67	111	3.74
Hard-Last-2	65	1.40	103	2.93
Hard-Last-1	47	1.21	88	1.89
fully latent (K=0)	37	1.18	51	0.93

observation text. The replay reproduces processed encoder tokens with a median ratio of 0.985 over 490 trajectories. Caching reduces encoder input from 1.87M to 25.7K tokens per trajectory, or from 1.65 PF to 0.02 PF. Deduplicating content-identical observations reduces it further to 11.0K tokens. A second seed gives 1.95M to 26.7K; an additional run gives 1.31M to 21.7K.

With per-observation caching, the replay estimates total LOHA compute at 2.12 PF per trajectory, compared with 1.82 PF for the Qwen3 base. This estimate concerns encoder reuse; the reported throughput was measured with the current serving implementation.

C Per-Seed Results and Recency-Window Sweeps

C.1 Sampling Repeats

Table 9 lists the individual runs for both model families. Means and standard deviations are computed from the main-evaluation seeds. The additional-run column provides reference results for the recency-window sweeps below.

C.2 Recency-Window Sweeps

The recency-window sweep in Table 3a is detailed in Table 10. Each condition uses one run, with one final result per instance.

For both models, resolve counts decrease substantially at K=1 and again at K=0. Increasing K from 3 to 8 raises Qwen3's resolved count from 68 to 72 and SWE-Master's from 111 to 115, while estimated decoder compute rises from 1.67 to 2.10 PF and from 3.74 to 8.37 PF per trajectory. Larger windows retain more text and generally improve task performance, although the results are not monotonic in K: Qwen3 resolves 76 tasks at K=4. The diagnostic ablation in Appendix B.3 separately compares K=0, K=3, and full text.

D IMPLEMENTATION AND TRAINING

This section gives the rendering, parameterization, and optimization details for LOHA + ACD. The objectives are defined in Section 2.3; their experimental analysis is in Appendix B.4.

D.1 CONTEXT RENDERING

Both training views use the decoder's native chat template and tool-call format. System prompts, tool schemas, the issue, and assistant turns remain in text. Compressed tool observations keep their original positions and response delimiters; their bodies are marked as memory regions:

```
<|im_start|>user
<tool_response>
<|memory_start|>... observation body... <|memory_end|>
</tool_response><|im_end|>
```

The 0.6B encoder processes each marked region in 1,024-token windows. The adapter maps it to $1/16$ as many soft tokens, while the surrounding tool envelope stays in text. No extra header is added: the preceding assistant tool call supplies the tool name and arguments.

In the latent training view, every observation body is encoded. In the full-text view, none is encoded. At inference, Hard-Last-K determines which observations are compressed, with an additional rule retaining bodies shorter than 128 characters in text. The original trajectory's assistant tokens and the appended read-back answers are the supervised positions.

D.2 STUDENT PARAMETERS

The student combines the base decoder with the LCLM encoder and mean-pooling MLP adapter. The encoder is frozen; the 9.2M-parameter adapter is trainable. The reading patch adds rank-ρ factors to every attention projection (q, k, v, o) and MLP projection (gate, up, and down): 252 matrices across 36 layers. Each factor pair is initialized from the rank-ρ truncated singular value decomposition of $\Delta W = W_{\mathrm{LCLM}} - W_{\mathrm{base}}$.

At rank 64, the patch has 132M parameters. We also train 144 decoder normalization gain vectors, initialized at their base values, for a total of 141.5M trainable parameters. All other decoder parameters stay frozen. Teacher distributions come from the unmodified base model and are cached before training. Recovering that decoder from the student requires removing the patch and restoring the original normalization gains.

D.3 LOSS AGGREGATION

Let $\mathcal{A}_{\mathrm{L}}$ and $\mathcal{A}_{\mathrm{H}}$ be the supervised assistant tokens in latent and full-text examples, and $\mathcal{R}$ the read-back answer tokens. For answer token j, let y_j be its target and c_j^{R} its preceding context. Let $\ell_{\mathrm{distill}}(j)$ and $\ell_{\mathrm{anchor}}(j)$ denote the per-token KL divergences in Equations 1 and 2, respectively. Then

$$\mathcal{L}_{\mathrm{distill}} = \frac{\sum_{j \in \mathcal{A}_{\mathrm{L}}} \ell_{\mathrm{distill}}(j) - \sum_{j \in \mathcal{R}} \log p_\theta(y_j \mid c_j^{\mathrm{R}})}{|\mathcal{A}_{\mathrm{L}}| + |\mathcal{R}|},$$

$$\mathcal{L}_{\mathrm{anchor}} = \frac{1}{|\mathcal{A}_{\mathrm{H}}|} \sum_{j \in \mathcal{A}_{\mathrm{H}}} \ell_{\mathrm{anchor}}(j).$$

The total loss is $\mathcal{L}_{\mathrm{distill}} + \lambda \mathcal{L}_{\mathrm{anchor}}$, with λ=1 in all reported runs. Each term is averaged over its own supervised tokens. The latent loss updates the adapter, patch, and gains; the full-text anchor updates only the patch and gains. Targets are aligned by assistant-token identity within a trajectory, since absolute positions differ between the rendered views.

D.4 CACHED TEACHER DISTRIBUTIONS

We score each full-text trajectory once and cache the teacher's top-M token log-probabilities and residual probability mass at every assistant position. The default is M=64; the full corpus contains

0.31B scored positions and uses 56 GB of cached targets. Each record is keyed by trajectory and assistant span and joined to the training examples during packing.

Both KL terms use the cached tokens plus one tail bucket. For teacher probabilities p_i, student probabilities q_i, residual teacher mass $r = 1 - \sum_{i \le M} p_i$, and student tail mass $q_{\text{tail}} = 1 - \sum_{i \le M} q_i$, the forward and reverse estimates are

$$D_{\rightarrow} = \sum_{i \le M} p_i \log \frac{p_i}{q_i} + r \log \frac{r}{q_{\text{tail}}},$$

$$D_{\leftarrow} = \sum_{i \le M} q_i \log \frac{q_i}{p_i} + q_{\text{tail}} \log \frac{q_{\text{tail}}}{\max(r, \rho_{\min})}.$$

Before numerical stabilization, coarsening gives lower bounds on the full-vocabulary divergences by the log-sum inequality. Equality holds when teacher and student have the same relative probabilities within the tail.

The early reference ablation uses top-16 targets, as marked in Table 7. Reverse-KL variants use $\rho_{\min}$=10^{-6} and clip per-token log-ratios at 10. The default recipe uses forward KL for both objectives. Approximation accuracy is measured in Appendix E.5.

D.5 Read-back Supervision

Latent-view trajectories include read-back questions at a density of one question per four observations. Questions are generated by tool type and ask for literal content such as a viewed code range, matching search lines, a failing test and assertion, a traceback frame, or an exit code. Candidate edit strings can also come from the trajectory's `old_str` arguments.

Candidates are filtered so that the answer is absent from hard-token context and cannot be inferred from the question text alone. Answer tokens receive next-token cross-entropy and account for 3% of supervised positions. The original trajectory tokens receive the distributional objectives above.

D.6 Data and Optimization

We select the shortest successful trajectory per task from CoderForge (Ariyak et al., 2026), giving 35,140 trajectories, 1.29B raw tokens, and 0.31B teacher-scored assistant positions. The Qwen3 diagnostic, teacher, and objective ablations use the first 3,514 trajectories under a fixed ordering. We apply no additional repository filter beyond the corpus's construction.

Both views are packed into 204,800-token sequences with span-level masks identifying distillation, anchor, and read-back supervision. The build checks that full-text rendering reproduces the teacher-scored token sequence; trajectories failing this check are dropped. Teacher arrays are checked against the student positions at load time.

We train for one epoch using AdamW with $\beta = (0.9, 0.95)$, a cosine schedule, and 10% warm-up. Learning rates are 10^{-4} for patch factors and normalization gains, and $5{\times}10^{-5}$ for the adapter. On the ablation corpus, one epoch consists of 125 optimizer steps of approximately 1.6M decoder tokens on eight GPUs. Every trajectory appears in both views during the epoch, and cached targets avoid a teacher forward pass during optimization.

D.7 Behavioral Probes

Candidate models are evaluated on the 96-field recitation and 48-field structured extraction probes, followed by a plain-text tool-calling probe on real tasks. The latter compares edits per trajectory and well-formed tool calls with the base model. It serves as a release criterion, but all reported ablation arms were evaluated regardless of their probe results. This separates reading quality from the ability to act with the scaffold's tools.

D.8 Inference and Caching

The agent uses the existing tool schemas and native function-calling harness. Before each call, LOHA keeps the last K observations and short bodies in text and marks older bodies for encoding. The

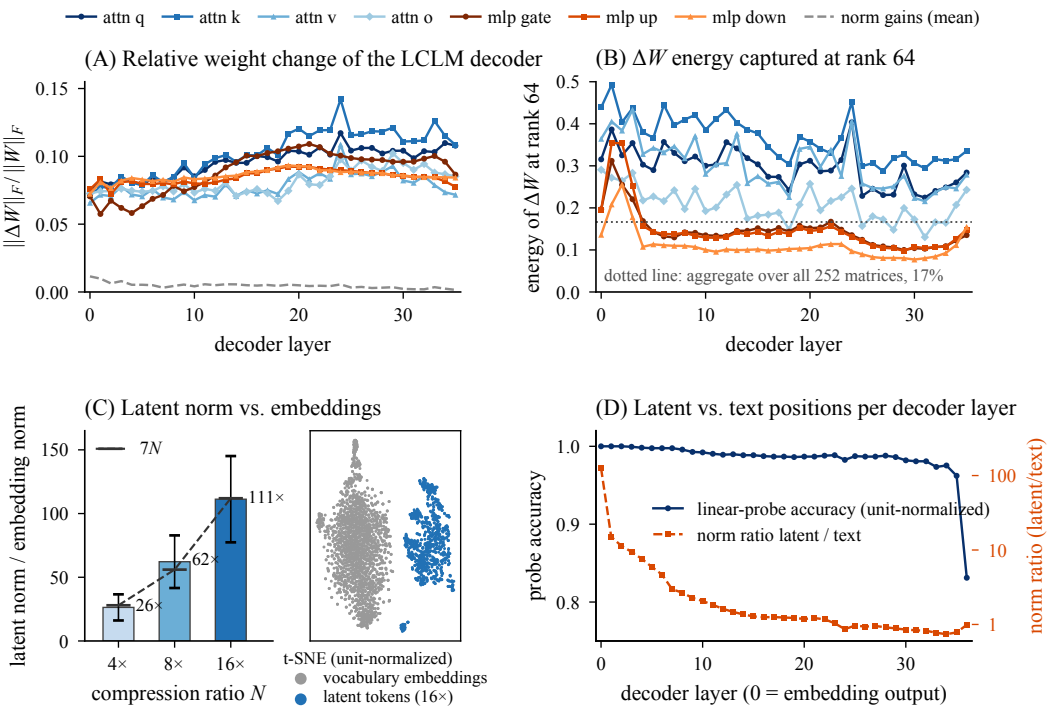


Figure 4: Measurements of the LCLM decoder and latent representations. (A) Relative Frobenius change of each projection matrix from the base to the LCLM decoder. (B) Fraction of the energy of ΔW in its rank-64 truncation. (C) Norm of the adapter's soft tokens relative to the word embeddings at three compression ratios, and a t-SNE of latent and vocabulary embeddings. (D) Per-layer accuracy of a probe separating latent from text positions, and the ratio of their norms.

server encodes those regions when the request arrives. The current implementation re-encodes them on every call; Appendix B.5 evaluates the savings from reusing encoder outputs.

Fully latent histories are append-only. Under Hard-Last-K, an observation changes from text to soft tokens when it leaves the window. This invalidates decoder prefix-cache entries from the first changed observation onward, so the recent tail and intervening assistant turns must be recomputed. The encoder and decoder cache effects are measured separately.

E Diagnostic Measurements

These measurements characterize the released LCLM decoder, its latent inputs, and the reading patch. They complement the agent-level ablations with direct tests of representations and recall.

E.1 Measurement Setup

Figure 4 compares the released $16\times$ LCLM decoder with Qwen3-4B-Instruct-2507. Encoder–adapter measurements use 48 windows of 1,024 tokens from real tool observations at compression ratios $4\times$, $8\times$, and $16\times$.

Panels A and B examine the 252 projection matrices. For $\Delta W = W_{\text{LCLM}} - W_{\text{base}}$, panel A plots $\|\Delta W\|_F / \|W\|_F$ and the mean change in four normalization gains per layer. Panel B reports the fraction of $\|\Delta W\|_F^2$ retained by rank-64 truncation; the aggregate is weighted by each matrix's squared change norm.

Panel C measures latent-token norms relative to mean vocabulary-embedding norm, with one-standard-deviation error bars. Its t-SNE uses 2,000 vocabulary embeddings and 1,000 latent embeddings after unit normalization, PCA to 50 dimensions, and perplexity 30. Panel D uses five-fold cross-validated

logistic regression to distinguish 512 latent positions from 1,152 text positions in unit-normalized residual-stream vectors at the embedding output and all 36 decoder layers.

E.2 Weight Changes and Reading Initialization

Projection matrices change by 6–14% of their Frobenius norm across layers. Normalization gains change by about 1% or less outside the first layer. The effective rank of ΔW ranges from 0.69 to 0.94 of full rank; rank-64 truncation retains 17% of total change energy, with a per-matrix median of 22%.

Applied to the base decoder, the rank-64 patch recovers 79% of the LCLM decoder's structured read-back score, while exact literal recitation reaches 27% of its rate. The same untrained patch degrades agent behavior. Thus the initialization recovers substantial reading ability, but its effect on acting still needs to be addressed during adaptation.

E.3 Latent and Text Representations

Mean adapter-output norms are $26\times$, $62\times$, and $111\times$ the word-embedding norm at $4\times$, $8\times$, and $16\times$ compression. Unit-normalized latent and vocabulary embeddings occupy separate regions in the t-SNE visualization. The linear probe distinguishes latent from text positions with 98–100% accuracy through layer 35, even after their norms have converged. These measurements show that latent and text positions remain distinguishable throughout most of the decoder.

E.4 Gradient and Teacher Diagnostics

At the warm-started patch, gradients of latent-view read-back and full-text next-token objectives have correlations of 0.93 and 0.95 in two measurements. A mask selecting directions with reading gain and low full-text cost retains 2,559 of 16,128 directions across the 252 rank-64 factor pairs. It recovers eight of 48 structured fields, against seven for the unpatched base and 22 for the full patch. The tested selection therefore removes most of the patch's reading gain.

Disagreement between the 30B teacher and 4B base model concentrates on 15.5% of positions with high-entropy next-action predictions. Their top-1 agreement is 39–65% at those positions and nearly complete elsewhere. The corresponding agent-level comparison appears in Appendix B.4, where behavioral anchoring and the distillation teacher are varied separately.

E.5 Cached-Distribution Approximation

On held-out positions at $M{=}32$, the cached forward-KL estimate is within 0.1% of exact KL. The reverse estimate underestimates by 3.5–4.2%, with gradient cosine 0.99; its gap concentrates on positions with student mass outside the cached tokens and approximately halves as M doubles.

E.6 Exact Recitation

We sample 358 literals that later actions quote in real trajectories, including `old_str` arguments, `pytest` identifiers, search patterns, and embedded code. The LCLM decoder receives the latent context and is scored on exact recitation, using the same sample at every compression ratio.

Accuracy is 20.4% at $4\times$, 15.6% at $8\times$, and 15.1% at $16\times$. No literal-type and length bucket with at least 10 examples reaches 90%. Composed `pytest` identifiers are never exact, and literals of at least 96 characters are almost never exact. Near-misses such as `description` rendered as `descript` illustrate why semantic access to an observation does not ensure exact tool arguments.

F Extended Related Work

F.1 Software-Engineering Agents

SWE-bench provides executable repair tasks from real GitHub issues (Jimenez et al., 2024). SWE-agent (Yang et al., 2024a), OpenHands (Wang et al., 2025), AutoCodeRover (Zhang et al., 2024), and Agentless (Xia et al., 2025) combine repository exploration, editing, and testing. CoderForge (Ariyak

et al., 2026) and SWE-smith (Yang et al., 2025) provide trajectories for training coding agents. Our work retains the tool interaction pattern and changes the representation of accumulated observations and the training needed to use it.

F.2 Context Management and Pruning

Observation masking can substantially reduce cost while retaining performance comparable to history summarization (Lindenbauer et al., 2025). Other approaches learn compression guidelines (Kang et al., 2026), fold completed sub-trajectories (Sun et al., 2025), restructure the workspace (Ye et al., 2025), page history through a hierarchy (Packer et al., 2023), or organize it as a dependency graph (Wu et al., 2026).

Methods targeting coding observations include SWE-Pruner's goal-conditioned line selection (Wang et al., 2026a), SWE-Pruner Pro's pruning head over coder hidden states (Wang et al., 2026b), TACO's evolved filtering rules (Ren et al., 2026), and CoACT's action-preserving compression targets (Chen et al., 2026). LOHA instead represents older observation bodies with soft tokens and keeps a recent text window. Masking supplies a direct comparison because it uses the same recency structure while removing the older content.

F.3 Prompt and Code Compression

Token-space compressors remove low-information tokens or spans (Li et al., 2023; Jiang et al., 2023; Pan et al., 2024). LongCodeZip uses function-level structure and conditional perplexity (Shi et al., 2025), while SlimCode simplifies code inputs independently of the downstream model (Wang et al., 2024). In an agent trajectory, a later edit or command may require an identifier or code fragment that was not salient when the observation arrived. LOHA separates recent exact text from a compressed history; the comparison with LongCodeZip examines this alternative on the same observation stream.

F.4 Latent Compression and Memory

Gist tokens (Mu et al., 2023), in-context autoencoders (Ge et al., 2024), and extreme compressors (Li et al., 2025) establish the interface between compressed embeddings and a language-model decoder. LCLM scales this approach with a 0.6B encoder and 4B decoder trained on 350B tokens (Li et al., 2026). We use that substrate for a multi-turn agent whose observations include code, search results, and test output.

Latent memory systems also encode GUI trajectories (Zhang et al., 2026), generate memory during reasoning (Zhang et al., 2025), or specialize it by agent role (Fu et al., 2026). These address experience reuse and coordination. LOHA focuses on observations within the current trajectory, including the exact text needed for coding-tool arguments.

F.5 Distillation and Behavioral Anchoring

Knowledge distillation transfers model predictions to a student (Hinton et al., 2015; Kim & Rush, 2016). Context distillation uses a richer teacher context to supervise a reduced student context (Askell et al., 2021; Snell et al., 2022); xRAG applies this idea to compressed retrieved documents (Cheng et al., 2024). ACD applies cross-view distillation at corresponding assistant tokens in tool-use trajectories. Work on teacher–student capacity gaps (Mirzadeh et al., 2020), imitation of stronger models (Gudibande et al., 2024), and same-capacity teachers (Furlanello et al., 2018) motivates examining the reference model explicitly. Reverse-KL and on-policy distillation offer alternative objectives (Gu et al., 2024; Agarwal et al., 2024).

The full-text anchor is related to Learning without Forgetting (Li & Hoiem, 2018), self-distillation fine-tuning (Yang et al., 2024b), and KL regularization toward a reference policy (Ziegler et al., 2019; Ouyang et al., 2022). Weight-space methods such as elastic weight consolidation instead constrain parameter changes (Kirkpatrick et al., 2017). ACD combines a distributional behavior anchor with latent-view distillation and trains a low-rank patch (Hu et al., 2022) initialized from the compressor's weight difference. The two objectives have distinct roles: one teaches the compressed representation, while the other regularizes the adapted agent toward its own base policy.